

# Course Concept Extraction in MOOCs via Embedding-Based Graph Propagation

Liangming Pan, Xiaochen Wang, Juanzi Li and Jie Tang

Knowledge Engineering Laboratory

Department of Computer Science and Technology

Tsinghua University, Beijing 100084, China

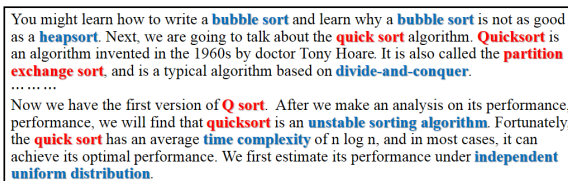
{panlm14@mails, wangxc15@mails, lijuanzi, tangjie}@tsinghua.edu.cn

## Abstract

Massive Open Online Courses (MOOCs), offering a new way to study online, are revolutionizing education. One challenging issue in MOOCs is how to design effective and fine-grained course concepts such that students with different backgrounds can grasp the essence of the course. In this paper, we conduct a systematic investigation of the problem of course concept extraction for MOOCs. We propose to learn latent representations for candidate concepts via an embedding-based method. Moreover, we develop a graph-based propagation algorithm to rank the candidate concepts based on the learned representations. We evaluate the proposed method using different courses from XuetangX and Coursera. Experimental results show that our method significantly outperforms all the alternative methods (+0.013-0.318 in terms of R-precision;  $p \ll 0.01$ ,  $t$ -test).

## 1 Introduction

In contrast with traditional courses that have limited numbers of students, each online course in a MOOC platform may draw more than 100,000 registrants (Seaton et al., 2014). The students have very diverse backgrounds; and knowledge presented in MOOCs is well understood by certain students, but might be difficult to others. In MOOCs, we use *course concepts* to refer to the knowledge concepts taught in the course videos, and related topics that help students better understand course videos. Identifying course concepts at a fine level is very important, as students with different backgrounds have totally different requirements. Figure 1 shows an example to illustrate the



You might learn how to write a **bubble sort** and learn why a **bubble sort** is not as good as a **heapsort**. Next, we are going to talk about the **quick sort** algorithm. **Quicksort** is an algorithm invented in the 1960s by doctor Tony Hoare. It is also called the **partition exchange sort**, and is a typical algorithm based on **divide-and-conquer**.  
.....  
Now we have the first version of **Q sort**. After we make an analysis on its performance, performance, we will find that **quicksort** is an **unstable sorting algorithm**. Fortunately, the **quick sort** has an average **time complexity** of  $n \log n$ , and in most cases, it can achieve its optimal performance. We first estimate its performance under **independent uniform distribution**.

Figure 1: Example of a video clip and its course concepts

problem addressed in this work. It is a clip (about ten minutes long) of video captions from the *data structure* course in XuetangX<sup>1</sup>, one of the largest MOOCs in China. The related course concepts are marked in red and blue. We see that there are a dozen concepts mentioned in this clip. The major concept is “quick sort” (marked in red), but the content is related to many other prerequisite concepts such as “divide-and-conquer”, “bubble sort”, and “unstable sorting algorithm” (marked in blue). For students with a computer science background, those concepts would be easy; however, for students from other discipline areas, the concepts may be completely new. With the identified course concepts, we can build an interactive learning environment to help individual students better grasp the knowledge in a course. For example, for a non-science student, the system may display the definition of “unstable sorting” and “uniform distribution”; while for a science student, the system could recommend advanced concepts or potential applications of “quick sort”.

Course concepts were previously provided by the teacher, but only at a coarse level—it is time-consuming and tedious to annotate all fine-grained concepts in all videos of a course. Our goal is to automatically identify all course concepts from each video clip. Despite quite a few studies on related research topics, including keyphrase extraction (Salton and Buckley, 1988; Mihalcea and Tarau, 2004; Liu et al., 2010) and term extrac-

<sup>1</sup><http://www.xuetangx.com/>

tion (Hisamitsu et al., 2000; Li et al., 2013), the problem of course concept extraction in MOOCs is far from solved. The most challenging issue in the MOOC context is the **low-frequency problem**. Course video captions often contain many course concepts with low frequency, primarily for three reasons: (1) course video captions are relatively short documents, containing small numbers of words; (2) many infrequent course concepts are from other prerequisite or related courses (e.g., “uniform distribution” is from mathematical courses and “divide-and-conquer” is from courses about algorithms); (3) a disambiguated course concept tends to be expressed in various ways, which produces many scattered infrequent terms. For example, “Q sort” is a colloquial expression referring to quick sort, and “partition exchange sort” is another name for quick sort. They both have infrequent presence in course videos.

Handling infrequency errors remains an open challenge for state-of-the-art keyphrase extractors (Hasan and Ng, 2014). In MOOCs, the low-frequency problem makes it difficult for video captions to provide reliable statistical information (e.g., *tf-idf*, *c-value*, and co-occurrence) for extracting and ranking terms, and results in the ignoring of many course-relevant yet infrequent concepts. For example, in Figure 1, we can correctly extract the frequent concept “quick sort” using *tf-idf*, but fail to extract “partition exchange sort” due to its infrequent presence, even if these two concepts have the same meaning. The above problem can be addressed if we know the semantic relationship between “quick sort” and “partition exchange sort”. Such an “understanding” can be facilitated by the incorporation of background knowledge from external data sources. There exist quite a few attempts (Vivaldi and Rodriguez, 2010; Rospocher et al., 2012) that utilize external knowledge sources such as Wikipedia categories and WordNet to incorporate semantic relations (e.g. synonymy, is-a) into keyphrase extraction. However, these methods are not suitable in the MOOC context, because many highly technical course concepts are not covered by generic semantic resources, including Wikipedia and WordNet.

Recently, distributed representations of words, namely word embeddings (Mikolov et al., 2013b,a), provides us with powerful tools to represent semantic relations between words. In our

work, to address the low-frequency problem, we incorporate online encyclopedias to learn the latent representations for candidate course concepts. Moreover, we present a novel graph-based propagation algorithm to rank the candidates based on the learned representations. We evaluate the proposed method using real courses from XuetangX and Coursera<sup>2</sup> with different domains and different languages, and compare our method with state-of-the-art keyphrase extractors. In summary, our contributions include: a) the first attempt, to the best of our knowledge, to systematically investigate the problem of course concept extraction in MOOCs; b) proposal of an efficient model for course concept extraction via semantic representation learning, and ranking candidates through graph-based propagation; c) design of four novel datasets using real courses of different disciplines from XuetangX and Coursera to evaluate our proposed method.

## 2 Problem Formulation & Framework

In this section, we first give some necessary definitions and then formulate the problem of course concept extraction.

A **course corpus** is composed of  $n$  courses in the same subject area, denoted as  $\mathcal{D} = \{\mathcal{C}_j\}_{j=1,\dots,n}$ , where  $\mathcal{C}_j$  is one course. We assume that course  $\mathcal{C}_j = \{v_{ij}\}_{i=1,\dots,m_j}$  consists of  $m_j$  course videos, where  $v_{ij}$  stands for the  $i$ -th video. Each video  $v_{ij}$  is composed of its video title  $t_{ij}$  and its video texts (video subtitles or speech scripts)  $d_{ij}$ .

**Course concepts** are subjects taught in the course. Formally, a course concept  $c$  can be defined as a  $k$ -gram in  $\mathcal{D}$  with the following properties: a) *phraseness*:  $c$  should be a semantically and syntactically correct phrase; b) *informativeness*:  $c$  should represent a scientific or technical concept related to courses in  $\mathcal{D}$ .

**Course concept extraction** is formally defined as follows. Given the course corpus  $\mathcal{D}$ , the objective is to extract candidate course concepts from  $\mathcal{D}$ , denoted as  $\mathcal{T} = \{c_1, \dots, c_M\}$ , and output the confidence score  $s_i$  for each candidate  $c_i \in \mathcal{T}$ .  $s_i$  indicates the likelihood of  $c_i$  to be a course concept in  $\mathcal{D}$ .

The fundamental challenge for course concept extraction is how to capture the phraseness and informativeness of course concepts. The general

<sup>2</sup><https://www.coursera.org/>

architecture of our model consists of three parts: (1) **candidate concepts extraction**, (2) **candidate concepts ranking**, and (3) **postprocessing**.

*Candidate concepts extraction* pre-processes the input corpus and extracts candidate course concepts. Considering that most concepts are noun phrases consisting of nouns, adjectives and some variants of verbs, and end with a noun word (Liu et al., 2010; Hulth, 2003; Li et al., 2013), we obtain candidate course concepts by extracting all noun phrases in the course corpus. The input text is first tokenized and annotated with part-of-speech (POS) tags. Next, we employ the linguistic pattern  $((A|N)^+|(A|N)^*(NP)?(A|N)^*)N$ , introduced by Justeson and Katz (1995), to extract all  $k$ -gram noun phrases as our candidates, where  $A$ ,  $N$ , and  $P$  denote the adjectives, nouns, and prepositions, respectively.

*Candidate concepts ranking*, which is the most important part of our method, involves ranking the extracted candidates based on their phraseness and informativeness. In our model, we utilize local mutual frequencies to measure the phraseness of the extracted candidates. The informativeness of a candidate is largely dependent on its semantic meaning. Due to the low-frequency problem, local statistical features do not provide sufficient information for capturing the semantic meaning carried by each candidate concept. In our method, we propose an embedding-based method, which incorporates external knowledge from online encyclopedias, to provide semantic representations for candidate concepts. Based on the learned phraseness and informativeness information, we construct a weighted undirected graph, namely a *course concept graph* (CCG), where each vertex represents a candidate course concept. We then rank the vertices in the graph via a novel iterative graph-based propagation algorithm, namely *course concept propagation* (CCP).

In the *postprocessing* step, by choosing an appropriate cut-off value for the ranking list, we can easily annotate course concepts in video texts via string matching.

### 3 Course Concept Graph

We combine the phraseness and informativeness information of candidate course concepts synergistically to construct the course concept graph, formally defined as follows.

The **course concept graph** (CCG) of  $\mathcal{D}$  is a

weighted undirected fully-connected graph denoted as  $\mathcal{G} = (V, E)$ , where  $V$  is the vertex set of  $\mathcal{G}$  and  $E$  is the edge set of  $\mathcal{G}$ . Each vertex in  $V$  represents a candidate course concept  $c_i \in \mathcal{T}$  and is associated with a *phraseness score*  $ph(c_i)$ , indicating the phraseness of  $c_i$ . For an edge  $(c_i, c_j) \in E$ , its edge weight  $e(c_i, c_j) = SR(c_i, c_j)$ , where  $SR(c_i, c_j)$  indicates the *semantic relatedness* (SR) between  $c_i$  and  $c_j$ , i.e., the likeness of their meaning or semantic content.

A fully-connected CCG stores the SR between any candidate pair. But in practice, we introduce a parameter  $\theta$  for pruning the graph. An edge  $(c_i, c_j)$  exists in a CCG only if  $SR(c_i, c_j) > \theta$ . The reason for pruning is: (1) the propagation on a fully connected CCG is computationally challenging, and (2) low-SR candidate pairs may introduce noise during propagation. In the following subsections, we introduce the essential building blocks of CCG, i.e., the calculation of phraseness and the representation of semantic relatedness.

#### 3.1 Phraseness Measurement

Phraseness examines the likelihood of a multi-word term to be a semantically and syntactically correct phrase. There exists several mechanisms such as Log-likelihood (LL) (Dunning, 1993) and Pointwise Mutual Information (PMI) (Church and Hanks, 1990) to measure the phraseness of a term. These methods are mostly based on the assumption that if the constituents of a multi-word candidate term form a collocation, rather than co-occurring by chance, it is more likely to be considered as a phrase (Korkontzelos et al., 2008). In our paper, we propose a simple PMI-based approach, which utilizes local mutual frequencies in the course corpus, to calculate the phraseness score for each candidate. Specifically, for a  $k$ -gram candidate  $c = \{w_1, \dots, w_k\} \in \mathcal{T}$ , where  $k > 1$ , we split  $c$  into  $f_i = \{w_1, \dots, w_i\}$  (prefix) and  $b_i = \{w_{i+1}, \dots, w_k\}$  (suffix), where  $i = 1, \dots, k-1$ . Then, the phraseness score  $ph(c)$  is defined as follows.

$$ph(c) = \max\{pmi(f_i, b_i) \mid i = 1, \dots, k-1\} \quad (1)$$

where  $pmi(f_i, b_i)$  is the PMI of the prefix  $f_i$  and the suffix  $b_i$ , defined as follows.

$$pmi(f_i, b_i) = \frac{2 \times freq(c)}{freq(f_i) + freq(b_i)} \quad (2)$$

where  $freq(c)$ ,  $freq(f_i)$ ,  $freq(b_i)$  are the occurrence frequencies of terms  $c$ ,  $f_i$  and  $b_i$  in the cor-

pus, respectively. Due to the low-frequency problem in the MOOC corpus, the phraseness scores for infrequent candidates may be statistically unreliable. Thus, the final phraseness of a candidate is estimated on both the MOOC corpus and the online encyclopedia corpus as follows.

$$ph(c) = \alpha \cdot F[ph^D(c)] + (1 - \alpha) \cdot F[ph^E(c)] \quad (3)$$

where  $ph^D(c)$  and  $ph^E(c)$  are the phraseness calculated on the course corpus and the encyclopedia corpus, respectively.  $\alpha$  ranges between 0 and 1, controlling the phraseness contribution of each corpus.  $F[\cdot]$  is a filter for the value of  $ph(c)$ . If the frequency of  $c$  is lower than a pre-defined threshold  $minc$ , or  $c$  is an unigram, the filter simply assigns a priori value of 0.5 to the term, i.e.,  $F[ph(c)] = 0.5$ .

### 3.2 Semantic Relatedness Learning

As mentioned in Section 2, statistical information in the course corpus cannot adequately capture the informativeness of concepts because of the low-frequency problem. In our method, we use word embeddings (Mikolov et al., 2013b,a) based on the online encyclopedia corpus to provide semantic representations for candidate concepts. Word embeddings represents each word as a low-dimensional, real-valued vector, and the semantic similarity between two words can be reflected by the cosine distance of their vectors. Our method for calculating semantic relatedness between candidate concepts consists of three steps: (1) corpus annotation, (2) representation learning, and (3) relatedness calculation.

**Corpus Annotation.** An online encyclopedia corpus is a set of articles, and can be represented as a sequence of words  $\mathcal{W} = \langle w_1 \cdots w_i \cdots w_m \rangle$ , where  $w_i$  denotes a word and  $m$  is the length of the word sequence. In  $\mathcal{W}$ , we replace any adjacent words which literally matches a candidate in  $\mathcal{T}$  as an unique token. After this step, we obtain a concept-annotated corpus  $\mathcal{W}' = \langle x_1 \cdots x_i \cdots x_{m'} \rangle$ , where  $x_i$  corresponds to a word  $w \in \mathcal{W}$  or a concept  $c \in \mathcal{T}$ . Note that the length  $m'$  of  $\mathcal{W}'$  is less than the length  $m$  of  $\mathcal{W}$  because multiple adjacent words are labeled as one token.

**Representation Learning.** We then learn word embeddings on  $\mathcal{W}'$  to obtain vector representations for all annotated candidates and words in  $\mathcal{W}'$ . For any unannotated candidate concept in  $\mathcal{T}$ , we

obtain its semantic representations via the vector addition of its individual word vectors.

**Relatedness Calculation.** When vector representations of candidate concepts are learned, we define the **semantic relatedness** between two candidates  $c_1$  and  $c_2$  as the cosine similarity of their vectors, denoted as  $SR(c_1, c_2)$ .

With this method, a candidate course concept has no corresponding vector only if any of its constituent word is absent in the whole encyclopedia corpus. This case is unusual because a large online encyclopedia corpus can easily cover almost all individual words of the vocabulary. Our experimental results verify that over 98% of the candidates have vector representations in this way.

## 4 Graph Propagation-Based Ranking

After the construction of a course concept graph, we propose an algorithm to rank the candidate course concepts, based on the following assumption about CCG. *In CCG, a course concept is likely to connect with other course concepts with high semantic relatedness.* We make this assumption based on two reasons. First, course concepts are usually scientific concepts. Scientific concepts in the same domain usually have strong semantic relations. Second, the main ideas of a document can be captured by a group or groups of words with strong semantic relationships. Course concepts are strongly related to the course corpus, which makes them more likely to connect with each other with high semantic relatedness.

For a vertex  $c$  in CCG, we denote  $conf(c)$  as its **confidence score** of being a course concept. Following the above assumption, the confidence score of a candidate course concept is evidenced by its semantic relations with other course concepts. In other words, high-confidence course concepts in CCG could propagate their confidence scores to their neighbor nodes that have high semantic relatedness with them, to discover other potential course concepts. Therefore, we propose an iterative graph-based algorithm, namely course concept propagation (CCP), which first assigns each vertex of CCG an initial confidence score, and iteratively updates the score for each vertex through propagation. The initial confidence score of each candidate course concept is determined by a **seed set**  $\mathcal{S}$ , which contains a list of known course concepts in  $\mathcal{D}$ . Specifically, let us denote the confidence score of the vertex  $c$  in the  $k$ -th iteration as

$conf^k(c)$ , and the initial confidence score of  $c$  as  $conf^0(c)$ . We set  $conf^0(c) = 1$  if  $c \in \mathcal{S}$  and  $conf^0(c) = 0$ , otherwise. In the following subsections, we first introduce the construction of a seed set, and then present the propagation process in detail.

#### 4.1 Seed Set Construction

In our method, the seed set can be constructed either manually or automatically. For some courses, key course concepts may already be included in course materials; however, this is not the case for most MOOC courses. For those courses in which key course concepts are not explicitly provided, we propose to extract them automatically from course outlines. A course in a MOOC usually contains hundreds of videos, each of which is associated with a video title. These video titles serve as an outline of a course and become a potential good resource for providing high-quality course concepts. Specifically, for each course  $\mathcal{C}_i$ , we first extract candidates from video titles of  $\mathcal{C}_i$ , following the same pattern-based procedure in the candidate extraction step. Then, candidates are ranked based on their *tf-idf* in  $\mathcal{C}_i$ . Finally, we select the top- $N$  ranked candidates to form the seed set  $\mathcal{S}$ .

#### 4.2 Propagation Process

To design the propagation process, we need to answer two crucial questions. First, a vertex should receive confidence scores from which vertexes in each iteration? Second, how much score should a vertex receives from another vertex in each iteration? Based on the above questions, we define two general functions as follows: a) **voting score**  $vs^k(c_j, c_i)$ : determines the confidence score  $c_j$  propagates to  $c_i$  in the  $k$ -th iteration; b) **voter-set**  $A(c_i)$ : it specifies the vertex set from which  $c_i$  receives the voting scores in each iteration. Therefore, a general propagation process can be defined as follows.

$$conf^{k+1}(c_i) = \frac{1}{Z} \left( \frac{\sum_{c_j \in A(c_i)} vs^k(c_j, c_i)}{|A(c_i)|} \right) \quad (4)$$

where  $Z$  a the normalization factor. The main idea of Equation 4 can be explained as a voting process. During each iteration, each vertex in  $A(c_i)$  will vote for  $c_i$ . The score of  $c_i$  in the next iteration is dependent on the average voting score of vertexes in  $A(c_i)$ . We implement the voting score

function as follows.

$$vs^k(c_j, c_i) = ph(c_j) \cdot e(c_i, c_j) \cdot conf^k(c_j) \quad (5)$$

where  $ph(c_j) \cdot e(c_i, c_j)$  determines the ‘‘authoritativeness’’ of the vote from  $c_j$  to  $c_i$ . If  $c_j$  has a higher phraseness (the voter is more credible) and  $c_j$  is more relevant to  $c_i$  (the voter knows more about the candidate), then  $c_j$  tends to propagate more of its score to  $c_i$  (the vote from  $c_j$  has a greater impact on determining whether  $c_i$  is a course concept than the votes from other voters). As for the implementation of  $A(c_i)$ , a natural way would be to set  $A(c_i)$  as the neighbor vertexes of  $c_i$  in CCG. Because a course concept is likely to connect with other course concepts with high semantic relatedness, it tends to receive high voting scores from its course concept neighbors, compared with other vertexes, during propagation.

However, in practice, the propagation process is usually hampered by the **overlapping problem**. If two terms  $c_i$  and  $c_j$  contain one or more identical words, we say that  $c_i$  and  $c_j$  are *overlapping*. For example, ‘‘merge sort’’ and ‘‘bubble sort’’ are overlapping because they both have the word ‘‘sort.’’ Overlapping frequently occurs among course concepts because scientific terms often have background words such as ‘‘function,’’ ‘‘algorithm,’’ and ‘‘culture’’. We find that  $vs(c_j, c_i)$  is less reliable if  $c_j$  and  $c_i$  overlap. For example, the term ‘‘same algorithm’’ is not a course concept, but it contains the word ‘‘algorithm,’’ which makes it to have a high SR with the course concepts including word ‘‘algorithm,’’ as in ‘‘bfs algorithm’’ and ‘‘kmp algorithm’’. Therefore, the score of the term ‘‘same algorithm’’ tends to be blindly increased by votes from these course concept neighbors. To solve this problem, we generalize the voting score by introducing a **penalty function**  $opf$ , which restricts the propagation of voting scores among overlapping terms. The **generalized voting score** ( $gvs$ ) is defined as follows.

$$gvs^k(c_j, c_i) = opf(c_i, c_j) \cdot ph(c_j) \cdot e(c_i, c_j) \cdot conf^k(c_j) \quad (6)$$

where  $opf(c_i, c_j) = 1$  when  $c_i$  and  $c_j$  is not overlapping, and  $opf(c_i, c_j) = \lambda$  otherwise, where  $\lambda$  ranges from 0 to 1.

**Termination Condition.** To determine when the iteration process stops, we introduce a **termination set**  $\mathcal{F}$ , including  $S$  course concepts. Let us denote the average ranking of concepts in  $\mathcal{F}$  after the  $k$ -th iteration as  $Ar^k(\mathcal{F})$ . The propagation process terminates if  $Ar^{k+1}(\mathcal{F}) > Ar^k(\mathcal{F})$ .

Dataset	Domain	Language	#courses	#videos	#tokens	#candidates	#labeled	correlation
CSEN	Computer Science	English	8	690	1,242,156	59,050	4,096	0.734
EcoEN	Economics	English	5	381	401,192	27,571	3,652	0.696
CSZH	Computer Science	Chinese	18	2,849	2,291,258	79,009	5,309	0.721
EcoZH	Economics	Chinese	8	455	645,016	60,566	3,663	0.646

Table 1: Dataset Statistics

## 5 Experiments

### 5.1 Datasets

Because there is no publicly available dataset for course concept extraction in MOOCs, we construct four course corpuses with different domains and languages to evaluate our model. Specifically, we collect Computer Science and Economics courses from two famous MOOC platforms — Coursera and XuetangX — to construct our evaluation datasets.

Coursera is one of the first MOOC platforms in the world. We collect video captions from 8 Computer Science courses to form the **CSEN** dataset. Similarly, video captions from 5 Economics courses are picked to construct the **EcoEN** dataset. All video captions in CSEN and EcoEN are in English. XuetangX is one of the most popular MOOC websites in China. Video captions of 18 Computer Science courses and 8 Economics courses are collected to form the **CSZH** dataset and **EcoZH** dataset, respectively. All video captions in CSZH and EcoZH are in Chinese.

The statistics of the four datasets are listed in Table 1, where *#courses*, *#videos*, and *#tokens* are the total number of courses, videos, and tokens in each dataset. For each dataset, the video captions are preprocessed following the procedure of *Candidate concepts extraction* in Section 2. *#candidates* denotes the number of extracted candidates for each dataset. To create a gold standard for evaluation, candidates for each dataset are sent to two human annotators majoring in the corresponding domain. For each candidate, the annotator is asked to make a judgement about whether it is a course concept based on the course contents. Thus, each dataset is doubly annotated, and Pearson correlation coefficient is applied to assess inter-annotator agreement. A candidate is labeled as a course concept only if the two annotators are in agreement. The column *#labeled* presents the ground-truth number for each dataset. In each dataset, we use 10% of ground truth to form the termination set and others for evaluation.

The datasets will be publicly available later.

As for the online encyclopedia corpus, we employ the Baidu encyclopedia<sup>3</sup>, which is the largest web-based encyclopedia in China, for Chinese language. Our training corpus includes 6,223,649 web pages crawled from the latest Baidu encyclopedia. For the English language, we extract 9,834,664 articles from the latest publicly available Wikipedia dump.<sup>4</sup>

### 5.2 Evaluation Metrics

For our experiments, we select two evaluation metrics. The first metric is **R-precision** ( $R_p$ ) (Zesch and Gurevych, 2009), which is an IR metric that focuses on ranking. Given a ranking list with  $n$  gold keyphrases, it computes the precision of a system over its  $n$  highest-ranked candidates. However, R-precision does not take the order of extracted keyphrases into account. To address the problem, we select the **Mean Average Precision** (MAP), which has been the preferred metric in information retrieval for evaluating ranked lists.

### 5.3 Influence of Parameters

We first investigate the parameters that may influence the performance including: (1)  $\alpha$  in Equation 3, (2) the size of seed set  $N$ , and (3) the penalty factor  $\lambda$ .

**The phraseness parameter  $\alpha$ .** The parameter  $\alpha$  controls the contribution of a course corpus when calculating the phraseness score. If  $\alpha$  is too large, the calculated phraseness may suffer from unreliable estimation; however, the calculated score may fail to reflect the real phraseness distribution in the course course when  $\alpha$  is too small. We investigate the influence of  $\alpha$  in Figure 2(a). This figure shows the R-precision of CCP when  $\lambda = 0.3$ ,  $N = 100$ , and  $\alpha$  ranges from 0 to 1. From this figure we find that, when  $\alpha$  is set from 0.2 to 0.7, the performance is consistently good, and remains stable with the variations of  $\alpha$ . When  $\alpha$  is larger than 0.85, the R-precision drops in all

<sup>3</sup><http://baike.baidu.com/>

<sup>4</sup><https://dumps.wikimedia.org/enwiki/20170120/>

evaluation datasets. The results demonstrate that the large-scale encyclopedia data is conducive to making reliable estimations of the phraseness. In our experiment,  $\alpha$  is set as 0.4 for all four datasets.

**The size of seed set  $N$ .** As mentioned in Section 4.1, we utilize the video titles to automatically construct the seed set for each dataset. We set the size of the seed set  $N = 10, 50, 100, 200$  for each dataset and explore the influences of  $N$  on CCP. Figure 2(b) shows the R-precision of CCP with different  $N$  on CSEH ( $\lambda = 0.3, \alpha = 0.4$ ). From the figures, we observe that the CCP reaches its best performance through 3 to 5 iterations. When  $N$  becomes larger, the algorithm tends to achieve the best performance faster, i.e., it terminates with fewer iterations. However, different settings of  $N$  all lead to a competitive best performance (around 0.43 on CSEH). We obtain similar observations on other datasets. In our experiments, we set  $N = 100$  for all datasets.

**The penalty factor  $\lambda$ .** The parameter  $\lambda$  reconciles the influence of voting scores from overlapping terms. We demonstrate the influence of  $\lambda$  in Figure 2(c). This figure shows that the CCP achieves its best R-precision when  $\lambda$  ranges from 0.20 to 0.40 in all datasets. If we do not severely lower the voting scores from overlapping terms ( $\lambda > 0.5$ ), or even have no penalty ( $\lambda = 1.0$ ), the performance is consistently unsatisfactory. The experimental results verify that the voting score from a non-overlapping neighbor (e.g., “data mining” to “machine learning”) is relatively more reliable than an overlapping one (e.g., “difficult learning” to “machine learning”) in CCP. However, voting scores from overlapping terms still encode some useful semantics (e.g., “merge sort” to “quick sort”), ignoring them ( $\lambda$  close to 0.0) may cause a decline in performance. We set  $\lambda = 0.3$  for all datasets.

## 5.4 Comparison with Baselines

After we explore the influences of parameters, we employ four baseline methods to compare with our proposed method, i.e., course concept propagation (CCP).

### 5.4.1 Baseline Approaches

First, we select two *statistics-based methods*, i.e., TF-IDF and PMI, as our baselines. **TF-IDF** ranks the candidate course concepts based on their *tf-idf* in the course corpus. For a candidate  $c \in \mathcal{T}$ , its *tfidf*  $R(c) = tf_c \times \log(idf_c)$ , where  $tf_c$  is the fre-

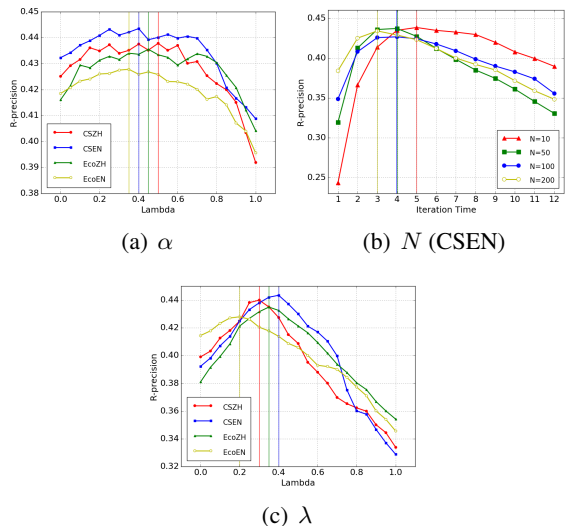


Figure 2: The Study of Parameter Influence on CCP

quency of  $c$  in  $\mathcal{D}$ , and  $idf_c$  is based on the number of videos in which  $c$  appears. In the **PMI** method, we directly rank candidates based on their phraseness, using the method described in Section 3.1. We use the same parameter  $\alpha$  with CCP for a fair comparison. Second, we also employ two *graph-based methods*. **TextRank** (Mihalcea and Tarau, 2004) and **Topical PageRank** (TPR) (Liu et al., 2010) are two state-of-the-art graph-based keyphrase extraction approaches. They all construct a graph based on word co-occurrences within the corpus, and apply the PageRank algorithm (Brin and Page, 1998) to score the vertices. The score of a multi-word candidate is the average score of all words within the candidate. Different from TextRank, TPR decomposes traditional PageRank into multiple PageRanks specific to various topics using LDA (Blei et al., 2003). Topics in TPR are learned from external data sources, and we use the same encyclopedia corpus as ours to make a fair comparison.

### 5.4.2 Implementation Details

For pruning the CCG, we set  $\theta = 0.3$  for all datasets. The *minc* is heuristically set as 5 in our experiments. The skip-gram model (Mikolov et al., 2013b) is applied to train word embeddings using the Python library *gensim*<sup>5</sup> with default parameters. For TextRank and TPR, we use the THUTag<sup>6</sup>, a state-of-the-art keyphrase extraction package, which provides the implementations of these methods.

<sup>5</sup><http://radimrehurek.com/gensim/>

<sup>6</sup><https://github.com/YeDeming/THUTag>

Method		CSEN	EcoEN	CSZH	EcoZH
TF-IDF	$R_p$	0.125	0.303	0.118	0.198
	MAP	0.105	0.232	0.109	0.145
PMI	$R_p$	0.239	0.222	0.246	0.179
	MAP	0.141	0.197	0.187	0.121
TextRank	$R_p$	0.151	0.290	0.142	0.161
	MAP	0.137	0.263	0.131	0.115
TPR	$R_p$	0.284	0.414	0.305	0.303
	MAP	0.255	<b>0.387</b>	0.267	0.288
CCP	$R_p$	<b>0.443</b>	<b>0.427</b>	<b>0.434</b>	<b>0.435</b>
	MAP	<b>0.432</b>	0.365	<b>0.416</b>	<b>0.423</b>

Table 2: Performance of Different Methods on Different Datasets

### 5.4.3 Performance Comparison

In Table 2 we summarize the results of different methods across different datasets. From the table, we find that the proposed methods outperform all baselines on all datasets, which indicates the robustness and effectiveness of CCP<sup>7</sup>. Specifically, we have the following observations.

First, TF-IDF performs competitively with TextRank, but they both perform worse than TPR and CCP. TF-IDF and TextRank only use statistical information within the course corpus and have a strong reliance on term frequency, which hampers their performances. For example, both TF-IDF and TextRank correctly extract the concept “IP”, which appears 139 times in the Computer Network course, but failed to extract “Internet Protocol”, the full name of “IP”, due to its infrequent presence. Moreover, they also highly ranked many irrelevant yet frequent terms, such as “English language”. These errors are significantly reduced in CCP with the incorporation of semantic relations.

Second, TPR performs better than TextRank across all datasets (an average of +0.141 in terms of R-precision), but is worse than CCP. In MOOCs, a course usually contain multiple topics, but TextRank often falls into a single topic, and fails to cover other substantial topics of a course. For example, in the Data Structure course, TextRank highly ranked phrases with “tree”, but lower-ranked concepts related to “sort”. TPR alleviates this problem by incorporating topic information from online encyclopedias. However, TPR also favors frequent course concepts, because frequent words tend to have high connectivity in the co-occurrence graph, and thus receive high rankings using PageRank.

<sup>7</sup>The improvements are all statistically significant tested with bootstrap re-sampling with 95% confidence.

## 6 Related Works

Our work is relevant to automatic keyphrase extraction, which concerns the automatic extraction of important and topical phrases from the body of a document” (Turney, 2000). Generally, keyphrase extraction techniques can be classified into two groups: supervised approaches and unsupervised approaches. In supervised machine-learning approaches, the training phase usually includes a classification task: each phrase in the document is either a keyphrase or not (You et al., 2013). Different learning algorithms have been employed to train the classifier, including naïve bayes (Frank et al., 1999; Witten et al., 1999), decision trees (Turney, 2000), maximum entropy (Yih et al., 2006; Kim and Kan, 2009) and support vector machines (Lopez and Romary, 2010; Kim and Kan, 2009). Unsupervised approaches usually involve assigning a saliency score to each candidate phrase, by considering various features (Wan and Xiao, 2008). Generally speaking, the information such as *tf-idf*, co-occurrence, or neighbor documents are frequently used in unsupervised keyphrase extraction. For example, TextRank (Mihalcea and Tarau, 2004) is a well-known method that ranks keywords based on the co-occurrence graph. Huang et al. (2006) utilize co-occurrence information to construct a semantic network for each document and derive the importance of phrases by analyzing the network. The ExpandRank (Wan and Xiao, 2008) model uses a set of neighborhood documents to enhance single-document keyphrase extraction. Recently, Liu et al., (2015) proposed a new framework that extracts quality phrases from text corpora integrated with phrasal segmentation. This model also rely on local statistical information and requires a relatively large corpus.

However, extracting low-frequency keyphrases remains an open challenge for all these methods. To address this problem, many related works consider not only the document level information but also knowledge from external data sources, to improve the effectiveness of automatic keyphrase extraction. Representative examples include the KEA++ system (Medelyan and Witten, 2006) that obtains candidate phrases from a domain-specific thesaurus. The work of Gazendam et al. (2010) also use the thesaurus as a background corpus. Besides the thesaurus, knowledge bases are widely used to calculate semantic relations between



terms. For example, Rospocher et al. (2012) use the WordNet to detect synonym terms, and then rank a synonym term higher even if it is ranked lower by the statistical method. The work of Vivaldi and Rodríguez (2010) utilizes an ontology hierarchy extracted from Wikipedia categories to provide background knowledge. Similarly, Berend and Farkas (2010) invent features concerning the Wikipedia level to achieve enhancements in performance. All these methods make use of only the explicit semantic knowledge contained in the external source. Different from them, we use word embeddings to learn semantic representations for candidate concepts and rank them via a novel graph-based propagation process.

## 7 Conclusion and Future Work

In this paper, we study the problem of course concept extraction in MOOCs. We precisely define the problem and propose a graph-based propagation method to extract course concepts by incorporating external knowledge from online encyclopedias. Experimental results on evaluation datasets validate the effectiveness of the proposed method. Incorporating external knowledge of various kind to help extract course concepts is an intriguing direction for future research. A straightforward task is to incorporate structured information such as “is-a” relation into the proposed model.

## References

- Gábor Berend and Richárd Farkas. 2010. SZTERGAK : Feature engineering for keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 186–189.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *International Journal of Machine Learning Research*, 3:993–1022.
- Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *International Journal of Computer Networks*, 30(1-7):107–117.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *International Journal of Computational Linguistics*, 16(1):22–29.
- Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *International Journal of Computational Linguistics*, 19(1):61–74.
- Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of IJCAI*, pages 668–673.
- Luit Gazendam, Christian Wartena, and Rogier Brussee. 2010. Thesaurus based term ranking for keyword extraction. In *Database and Expert Systems Applications, Dexa, International Workshops, Bilbao, Spain, August 30 - September*, pages 49–53.
- Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*, pages 1262–1273.
- Toru Hisamitsu, Yoshiki Niwa, and Jun’ichi Tsujii. 2000. A method of measuring term representativeness - baseline method using co-occurrence distribution. In *Proceedings of COLING*, pages 320–326.
- Chong Huang, YongHong Tian, Zhi Zhou, Charles X. Ling, and Tiejun Huang. 2006. Keyphrase extraction using semantic networks structure analysis. In *Proceedings of ICDM*, pages 275–284.
- Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, pages 216–223.
- John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.
- Su Nam Kim and Min-Yen Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the ACL-IJCNLP Workshop on Multiword Expressions*, pages 9–16.
- Ioannis Korkontzelos, Ioannis P. Klapaftis, and Suresh Manandhar. 2008. Reviewing and evaluating automatic term recognition techniques. In *Proceedings of GoTAL*, pages 248–259.
- Sujian Li, Jiwei Li, Tao Song, Wenjie Li, and Baobao Chang. 2013. A novel topic model for automatic term extraction. In *Proceedings of SIGIR*, pages 885–888.
- Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of ACM SIGMOD*, pages 1729–1744.
- Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376.
- Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 248–251. Association for Computational Linguistics.

- Olena Medelyan and Ian H. Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of JCDL*, pages 296–297.
- Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP*, pages 404–411.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *International Journal of CoRR*, abs/1301.3781.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.
- Marco Rospocher, Sara Tonelli, Luciano Serafini, and Emanuele Pianta. 2012. Corpus-based terminological evaluation of ontologies. *Applied Ontology*, 7(4):429–448.
- Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *International Journal of Information Processing and Management*, 24(5):513–523.
- Daniel T. Seaton, Yoav Bergner, Isaac L. Chuang, Piotr Mitros, and David E. Pritchard. 2014. Who does what in a massive open online course? *Commun. ACM*, 57(4):58–65.
- Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *International Journal of Information Retrieval*, 2(4):303–336.
- Jorge Vivaldi and Horacio Rodríguez. 2010. Finding domain terms using wikipedia. In *Proceeding of LREC 2010*.
- Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AACL*, pages 855–860.
- Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Kea: practical automatic keyphrase extraction. In *ACM Conference on Digital Libraries*, pages 254–255.
- Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of WWW*, pages 213–222.
- Wei You, Dominique Fontaine, and Jean-Paul A. Barthès. 2013. An automatic keyphrase extraction system for scientific documents. *Knowl. Inf. Syst.*, 34(3):691–724.
- Torsten Zesch and Iryna Gurevych. 2009. Approximate matching for evaluating keyphrase extraction. In *Proceedings of RANLP*, pages 484–489.