TakeThis: 도메인에 특화된 개념의 유사도 기반 대학 교과목 추천 시스템

손병호^{O} 이태한 구명완
서강대학교 컴퓨터공학과
bhson99@sogang.ac.kr, sogangcse@sogang.ac.kr, mwkoo@sogang.ac.kr

TakeThis: A University Course Recommendation System Based on Domain-Specific Concept Similarities

Bjongho Son^{O} TaeHan Lee Myoung-Wan Koo
Department of Computer Science and Engineering, Sogang University

ABSTRACT

When measuring the semantic similarity between two concepts, word2vec is a typical choice. However, word2vec inherently possesses a trade-off between training efficiency and semantic expressivity. Our approach overcomes this trade-off by viewing a concept as a structured collection of relevant keywords rather than a natural language word. We evaluate our approach by implementing our university course recommendation system, *TakeThis*. The result of *TakeThis* is 4.65 times more accurate than the theoretical baseline.

1. INTRODUCTION

When measuring the semantic similarity between two concepts, one possible way is to view the concepts as natural language words and compute the similarity between two word2vec vectors. word2vec methods such as CBOW[1], skip-gram[2] are usually given as pretrained embeddings, trained on large public corpora such as Wikipedia. However, these public corpora are highly likely that they don't contain domain-specific concepts (e.g., technical terms, made-up words, etc.), so that word2vec fails to capture the domain-specific semantics. One could handle this issue by fine-tuning the word embeddings by using private corpora containing domain-specific words. However, this is too costly in that it requires additional training and corpora. Hence, word2vec inherently possesses a trade-off between training efficiency and semantic expressivity.

Our approach overcomes the trade-off by viewing a concept as a structured collection of relevant keywords – which we call a *signature*, rather than a natural language word. Provided that a corpus[1] for each domain-specific concept is available, our approach uses keyword extractor to encode the domain-specific concepts as *signatures*.

We demonstrate the effectiveness of our approach by implementing a university recommendation system *TakeThis*, in which the measurement of similarities between university courses (domain-specific concept) and a career (general concept) is included.

2. RECOMMENDATION SYSTEM

2.1 Overview

Problem Formulation. Given k and a career name from the user, the goal of the recommendation system is to return top-k relevant courses for the given career.

For example, if the user gives "graphics engineer" and $k = 5$ as input, the system should return 5 courses, possibly including courses such as *Introduction to computer graphics*. In order to find most relevant courses for the given career name, it is crucial that we accurately

[1] This research was supported by the MSIT (Ministry of Science ICT), Korea, under the National Program for Excellence in SW (2015-0-00910) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

measure the semantic similarities between two concepts, namely careers and courses. Our approach is to encode these concepts as vectors by keyword extraction and measure their similarities by simply computing the similarities between vectors. This may seem like a typical word embedding in that a concept is encoded into a vector. However, our approach does not require any learning phases to encoding the concepts. Furthermore, the concepts to be encoded may have domain-specific semantics - the semantics of a course name is tightly coupled with a particular university, a professor, etc. For this, we maintain our own data structures - the glossary, course signatures, and the career signature.

Definition 1. The glossary is a set of keywords, obtained by collecting all the keywords extracted by applying the keyword extractor on the corpus for each course, where each corpus contains the description for the corresponding course.

Definition 2. A signature for a concept (i.e., course/career) is a vector of size $n = |glossary|$, where the *i*-th value indicates how relevant the *i*-th keyword in the glossary is to the concept of interest, provided that the glossary is ordered appropriately.

By encoding concepts as signatures, we can canonically compute their distances by typical vector operations. Our signature encoding process is done in three stages. First, we build the glossary out of course corpora, thereby encoding all courses as signatures. Second, with the glossary, we encode the career given by the user. Third, we measure the course-career similarities by computing the similarities between their signatures, then take top-k courses as output. The process is depicted in Figure 1. We explain each step in more detail.

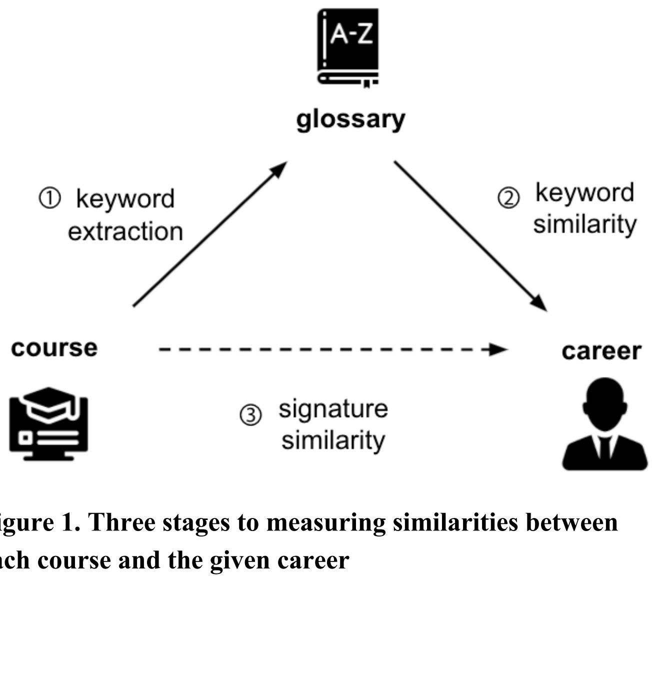

Figure 1. Three stages to measuring similarities between each course and the given career

2.2 Keyword Extraction

To encode a course into a signature, we apply a keyword extractor on the corresponding corpus for the course. By doing so, we obtain p pairs each of form (*keyword*, *score*), where p is a hyperparameter indicating the number of keywords to be extracted. For example, keyword extractor on corpus for the course *database systems* with $p = 5$ gives (*"database"*, 0.505), (*"systems"*, 0.3382), (*"differences"*, 0.323), (*"data"*, 0.317), and (*"architectures"*, 0.315). Then the set obtained by collecting every keyword that occurs at least once in the result of each course, becomes our glossary. With this glossary, we can encode a course into a signature, by setting the scores of each entry in the signature to be the scores from the keyword extractor. Note that a course signature is a sparse vector, since all but p entries have value zero.

2.3 Keyword Similarity

The signature of the given career is in exactly the same format as those of courses. However, we can't use the keyword extractor to encode it into a signature, as we did for the courses. This is because unlike courses, we have no fixed list of all possible careers and the corpora for that matter, since a user may give an arbitrary career name as input. Thus, we use a different method to build the career signature. In particular, the *i*-th value of the career signature is set to be the similarity score between the career name and the *i*-th keyword in the glossary. We define our own similarity measure sim_kw in definition 4, to compute the similarity score.

Definition 3. The document function $docs : string \rightarrow 2^{Doc}$ is defined by $docs(str) = \{d \in Doc |\ d\ contains\ str\}$, where Doc is the set of all documents.

Definition 4. The keyword similarity function $sim\_kw : string \times string \rightarrow [0,1]$ is defined by

$$sim\_kw(kw_1, kw_2) = \frac{|docs(kw_1) \cap docs(kw_2)|}{|docs(kw_1) \cup docs(kw_2)|}$$

In practice, we estimate the output of $docs(str)$ by the number of results obtained by web searching with query being str. The intuition behind definition 4 is that the more similar two keywords are, the more documents containing both keywords exist, ignoring the effect of the frequency of a single keyword per se.

2.4 Signature Similarity

Now that we have signatures for each course and the signature of the given career, measuring similarities between each pair is a trivial step. We use a typical vector similarity measure, which is cosine similarity, as our signature similarity measure.

Definition 5. The signature similarity function $sim\_sig: [0,1]^n \times [0,1]^n \rightarrow [0,1]$ is defined by

$$sim\_sig(sig_1, sig_2) = \frac{\langle sig_1, sig_2 \rangle}{\|sig_1\| \cdot \|sig_2\|}$$

, where $n = |glossary|$.

With this measure, taking the top-k most similar courses against the given career gives the final output of our system.

3. EXPERIMENT

3.1 Experimental Setup

We evaluate our system by implementing *TakeThis*, now available at [3]. We share our whole experiment by the video demo on [4]. For simplicity, we restrict our attention to only CSE (Computer Science and Engineering) courses and CSE careers. We set $k = 5$ (number of courses) and $p = 5$ (number of keywords for keyword extraction) as our hyperparameters.

Our system is based on the following external sources:
(1) Corpus: For each course, we collected the corpus from [5]. It contains 58 course descriptions provided by the CSE department of Sogang University.
(2) Keyword Extractor: We used *KeyBert*[6], which is based on *BERT*, as our keyword extractor described in section 2.2.
(3) Document Function: We used Google Custom Search API[7] as our document function $docs$ described in definition 3. We restrict the documents to be from one of *stackexchange.com, ziprecruiter.com, hired.com, indeed.com, careerexplorer.com, quora.com*

3.2 Evaluation

We evaluate the effectiveness of our system by giving representative CSE careers as inputs and compare the output courses against the test data. The result is shown in Table 1. To measure the plausibility of the result, we established a carefully annotated test data which consist of 5 most relevant courses for each career. Then, we compute the average hit ratio of our system. We define the hit ratio for a career to be the accuracy of our 5 recommendations with respect to the test data. Our result yields the average hit ratio $r = 0.4$. By simple combinatoric calculations, the hit ratio of random 5 recommendations from 58 courses yields $r = 0.086$. With this theoretical baseline, our system shows 4.65 times better accuracy.

Table 1. Result of TakeThis for three selected careers. The hit course is denoted by (*). Score indicates the cosine similarity between a career and a course.

Career	Course Name	Score
Graphics Engineer ($r = 0.4$)	C programming	0.373
	Intro. to Computer Graphics (*)	0.074
	Windows Programming (*)	0.055
	Internship	0.054
	Operating System	0.025
Data Scientist ($r = 0.4$)	C programming	0.376
	Database Systems (*)	0.069
	Data Mining (*)	0.069
	Programming Languages	0.049
	Intro. to Data Communication	0.046
Hardware Engineer ($r = 0.4$)	C programming	0.370
	Basic SoC Design (*)	0.148
	CSE Laboratory I	0.140
	Intro. to CSE	0.059
	Computer Architecture (*)	0.049

4. CONCLUSION

We propose *TakeThis*, a university course recommendation system based on lightweight measuring of similarities for domain-specific concepts. Our empirical study shows that *TakeThis* is 4.65 times more accurate than the theoretical baseline. We address that our approach is not restricted only to our specific application but is broadly applicable to many problems that requires approximating the semantics of domain-specific concepts, when a corpus for each concept is available.

REFERENCE

[1] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[2] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.

[3] https://github.com/byhoson/takethis

[4] https://youtu.be/i2J0_EAGYWsAdsf

[5] https://ecs.sogang.ac.kr/ecs/ecs03_2_1.html

[6] https://maartengr.github.io/KeyBERT/

[7] https://developers.google.com/custom-search/v1/overview