École polytechnique fédérale de Lausanne
Center for Digital Education

Calculating Similarity Between EPFL Courses and Using It to Build a Graph

Master's Project Report

January, 2019

Robert Injac

Supervisors:
Francisco Pinto
Patrick Jermann

1 Introduction

The university École polytechnique fédérale de Lausanne (EPFL) has more than one thousand courses in various disciplines. Navigating through those courses can be difficult. Especially hard is to find courses relevant to a certain course: the pre-requisites for a certain course, or courses to further knowledge in a field if a person has finished a certain course.

In this project, I developed a method for establishing similarity/relatedness[1] between two EPFL courses. The method is based on natural language processing (NLP) methods, particularly the word embeddings (which will be explained later). The method returns a real number between 0 and 1, with larger number indicating greater similarity between two courses.

I use this method to construct a graph of related EPFL courses. In the graph, the nodes are the courses, and there is an edge between the two nodes if those courses are related.

2 Data

There are 4 datasets that are used in this project:

- **Course descriptions:** contains course name, description, summary, and other data for all EPFL courses
- **Course dependencies:** contains some EPFL courses for which professors said are similar (used for testing the method)
- **Course keywords:** contains keywords for each EPFL course (keywords are usually main concepts taught in the course)
- **Course semesters:** contains in which semester is each course mostly taken (for example, Master 2. semester)

The data comes from EPFL course databases. Some of the data is corrupted: usually, course name or description is missing. Another possibility is the language mix-up: sometimes, there is a title in French where it should be in English and vice-versa. I tried to fix these inconsistencies as much as possible: I fixed most cases of language mix-up, but if crucial data (such as course code) is missing from an entry, I discarded it.

[1] I will use the terms similar courses and related courses synonymously, since I assume that the notion of course "similarity" is the same as course "relatedness".

3 Pre-processing

Pre-processing is a vital step in any NLP method. For each EPFL course there is the following data:

- **Course name:** name of the course.
- **Course description:** around 50-100 words, description about what is the course about.
- **Course summary:** summary of the curriculum of the course
- **Course keywords:** important concepts learned in the course.

I will use data obtained from all 4 in my method. So, for each course I have made a "course content" field which is just all 4 of that, tokenized and then concatenated into a single word list. I apply standard pre-processing on this word list: convert to lowercase, removing stop words, removing punctuation.

In the pre-processing step, I also remove top 100 most common words found in all courses. I want each course content list to have as much as possible words related to that course, and the least possible amount of words which are not related to that course. Considering a lot of words are related to all courses (such as "course", "project", "theory", etc), these words generally increase the similarity scores by making all courses more similar to each other, because they all share those words. And even worse, if some course has a smaller proportion of these common words, it will seem less related to other courses just because of that. So I fix this by removing the top 100 most common words.

4 Similarity Method

4.1 Word embeddings method

Word embedding is the name for a set of NLP techniques in which words or phrases from the vocabulary are mapped to vectors of real numbers. It's a projection from a space with one dimension per word to a continuous vector space with a much lower dimension. One of the most used word embeddings method is called *word2vec*[2].

My method uses word embeddings. The word embeddings I chose are *fasttext word2vec* embeddings: 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and *statmt.org* news dataset (16B tokens)[1]. These pre-trained embeddings are freely available online[2].

The similarity method is simple. To establish similarity between two courses, **both courses are projected into the vector space and the cosine similarity is found between them.**

[2] `https://fasttext.cc/docs/en/english-vectors.html`

To project the course into a vector space, the word embeddings are used. There is a list of 200 to 1000 words related to the course for each course (the course content field we got from pre-processing step). I get the *word2vec* word embedding vector for each word in that list. Then I take the average of those word embedding vectors, and that is the embedding of the course in the vector space.

Why should this work? If two words are semantically similar, their word embedding vectors should be close to each other in the vector space. The words in the course content list should mostly be related to the course. If we take the "course embedding" for each course (which is calculated from that list of words), it should be that similar courses are near to each other in vector space: most of their words in course content list should be semantically similar.

4.2 Evaluation

To check if my method is working, I wanted to evaluate it. I have the ground truth (For some courses, I know which courses are truly similar to each other - the course dependencies dataset), so I will compare my method's results with those.

In the previous section, I established a method for measuring the similarity of two courses. But this method gives me a continuous value between 0 and 1 - and I want to have a discrete value: related or not related. I need one hyperparameter: threshold. If the similarity is above the threshold, the courses are related, and if it is below, they are not.

To calculate the best threshold I have used cross-validation. I made an evaluation dataset consisting of 172 similar and 250 not-similar courses. I have split the dataset: 70% for the train set and 30% for the test set. I have found the best value for my hyperparamater, the threshold, on the train set. It is the threshold for which the F-score is the highest. Then, I have evaluated my method on the test set.

Accuracy	Precision	Recall	F1-score
0.81889	0.77777	0.79245	0.78504

Table 1: Results of the evaluation

The results of the evaluation are seen on table 1. As can be seen from the results, the method manages to predict whether the two courses are related quite well.

5 Graph Creation

In the previous section, I have shown how I established and evaluated the method for calculating similarity/relatedness between two EPFL courses. In this section, I will show how I created a graph of EPFL courses. I wanted to create a graph in such a way that two related courses share a link.

The graph has the following structure:

- **Nodes:** EPFL courses
- **Edges:** two courses will share an edge if they are related (according to our similarity method)

I went through all the courses and found similarity between each pair. For each course, I connected it to the three most similar courses, provided that the score for each is above a certain threshold (I kept the threshold from the evaluation at 0.88). This is how I built the graph.

The graph is **directed**. I have the information which courses are taken when (course semesters dataset), and I only connected courses from earlier to later.

5.1 Visualization

To visualize the graph, I used the work of an EPFL student Raphaël Steinmann. His project was visualizing the graph of EPFL courses; he did it using D3.js JavaScript framework[3]. I took his code for visualization, available at GitHub[3], and put the data of my graph in it.

The interactive visualization is available online, hosted on GitHub pages[4]. The visualization of the whole graph is shown on figure 1. Selecting one course, the similar courses of that course are shown, as can be seen on figure 2.

There is no ground truth for all data, so I cannot evaluate are the courses connected on the graph really relevant to each other. However, using the interactive visualization, one can click around and try to "check" if the connected courses are relevant. In most cases, they do seem relevant, but there are cases where the graph is clearly wrong.

6 Conclusion

In this project I established the similarity method and built the graph. While the similarity method was evaluated, the graph cannot be evaluated since we don't know the ground truth for all data.

The similarity method used is quite simple. It is possible that better results can be obtained using a more complex method.

The entire repository of the project is available at GitHub[5]. In the repository there are the used datasets and two notebooks: one for similarity method, and one for graph creation. The notebooks contain descriptions, explanations and comments along with the code.

[3] `https://github.com/rbsteinm/EPFL-Courses-Graph-Representation`
[4] `https://robertinjac.github.io/EPFL-Courses-Similarity/`
[5] `https://github.com/RobertInjac/EPFL-Courses-Similarity`

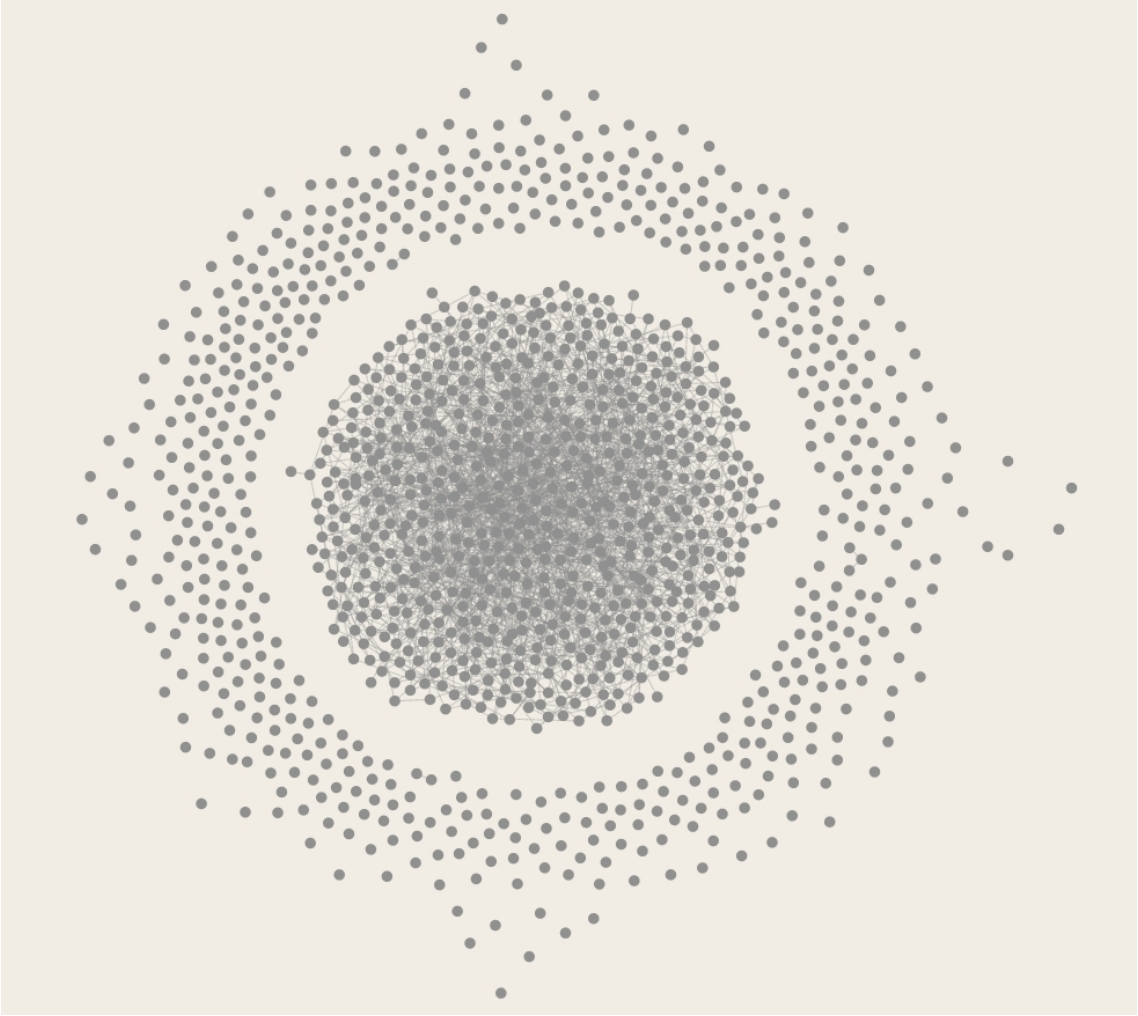

Figure 1: Graph visualization

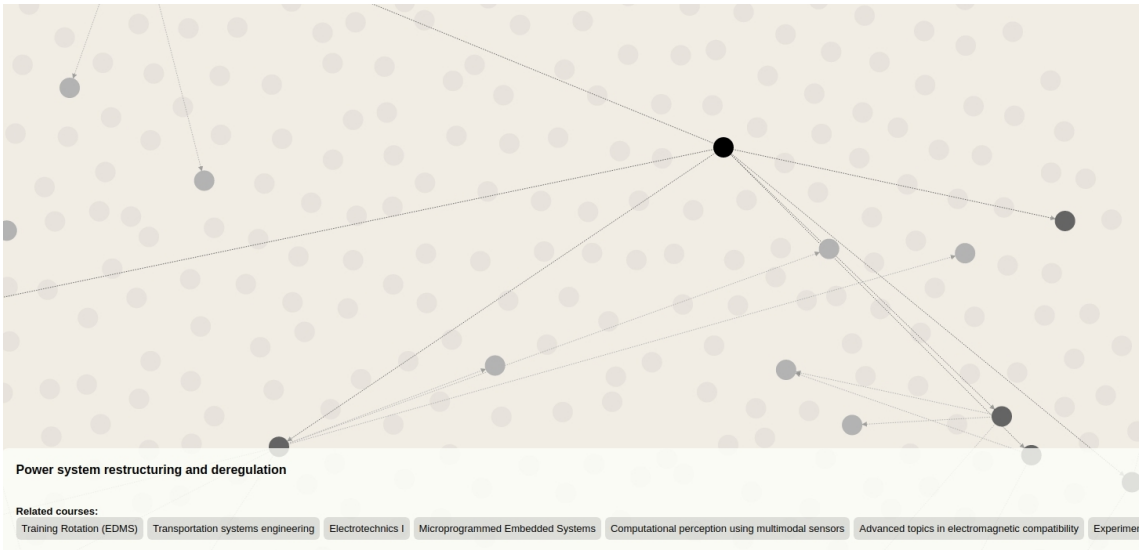

Figure 2: Selecting one course

References

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Raphael Steinmann. Directed graph visualization of epfls course catalogue using d3.js. 2018.