

MOOCer 2.1

CS 6365 Spring 2020 Final Report

Team members: Xin Zhang, Shuangke Li

I. MOTIVATION

Massive open online course (MOOC) is an online course aimed at unlimited participation and open access via the web. This new form of education is getting more and more popular nowadays. Compared to traditional forms of class, online courses provide a lot of benefits such as easy accessibility, high variability and low cost. There are a lot of platforms host online courses such as edX, Udacity and Coursera. As these platforms keep publishing new courses, search of online courses across multiple platforms become challenging. EdX, for example, alone has 2650 courses by 2019. In addition, some people only have general career goals such as becoming Software Developer. Because they are not familiar with the career field and lack understanding of skillsets required by the job, they are unable to determine what courses to search for. In addition, job market is changing, some online courses are useful now for certain jobs may not be useful years later. Those problems encourage us to develop an website which could help people find the suitable courses across multiple online education platforms to boost their career goals. By using our website, we wish users could save time in course searching and learn knowledge more efficiently with courses highly related to their career interest.

II. PROJECT ARCHITECTURE

Figure 1 gives overall architecture of our project. We make API calls to collect data from Job information sources like CareerBuilder and online course information sources like edX. Updated data is used to build and update course Matching model. The course matching model takes job cluster id as input and output list of matched courses. When users type in keyword of the job in our website, our keyword matching algorithm will match keyword to existing job cluster id. The job cluster id will then be passed to course matching model as input. The website will then display course list from course matching model output.

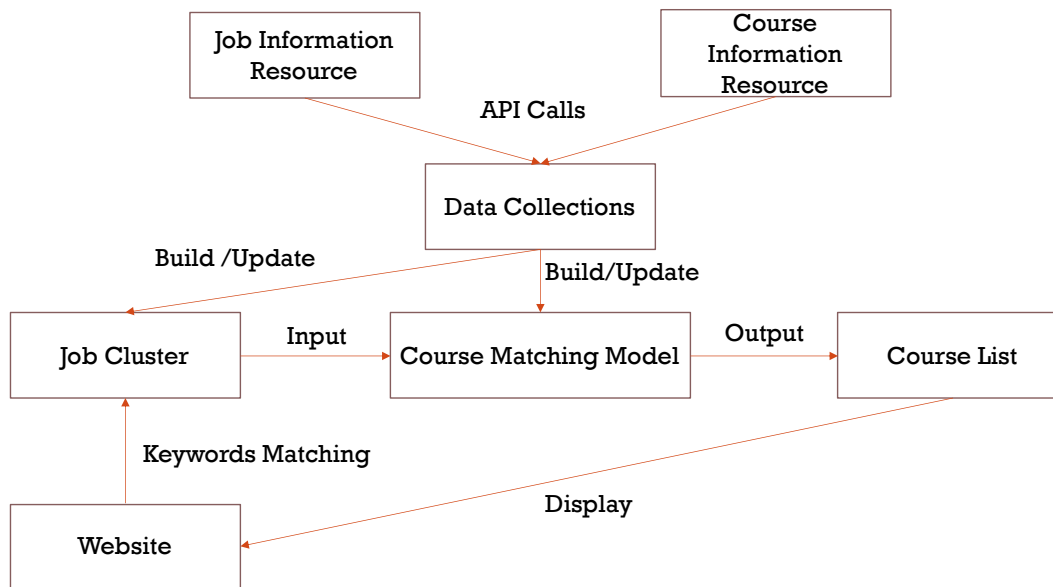


Figure 1

1. DATA COLLECTION AND CLEANING

16048 job posts have been collected from online career sources including Career Builder, Google Career Site, Yahoo Career Center. Job Title, Job Description and Job Requirement are extracted from each job posts. 12084 courses information have been collected from online courses platforms including edX, Udacity and Coursera. Course Title, Courses Description and Courses URL have been extracted.

Job Descriptions, Job Requirements and Courses Descriptions have been preprocessed by removing urls, cleaning punctuations and special characters, removing HTML tags and correcting spell errors.

2. COURSE MATCHING MODEL

We build our course matching model in three steps: cluster similar jobs; extract key words from job requirements and course descriptions; matching courses and job clusters based on key word Lists.

2.1 Job Clustering

There are two reasons we do job clustering. First of all, many jobs have similar functionalities and responsibilities but have different titles in different companies. For example, some company may use software engineer, others may use software developer or software development engineer etc. The second reason is that when we retrieve key words from job requirement, job clustering could help us filter out unpopular key words from job requirements with majority vote. For example, one

software engineer job may require English skills but this skill is not a popular skill if we look at the entire cluster of software engineers and we could filter out that key word as job requirement.

We used Glove(Global Vectors for Word Representation) which is a pre-trained word representation to do word embeddings on job descriptions. Each word is embedded into 200 dimension vector and we created N by 200 matrix for job descriptions. Then we use DBSCAN (Density-Based Spatial Clustering of Applications with Noise) for clustering vectorized job descriptions. We picked DBSCAN as our clustering algorithm because we don't know how many clusters we have in our job lists. Other popular clustering algorithms like K-means or Spectral Clustering Algorithm need the number of clusters as input parameter. Figure 2 shows an example of one job cluster. Jobs in this cluster have variations in job titles but have similar functionalities and responsibilities.

Cluster ID 2841:

Job ID:	Job Title:
3180	C++ Senior Software Developer (UNIX)
3338	C++ Senior Software Developer
3519	C++ Senior Software Developer (UNIX)
3811	C++ Senior Software Developer
4046	C++ Senior Software Developer
4504	C++ Software Developer
4669	C++ Senior Software Developer
4946	C++ Senior Software Developer
5049	C# Software Developer
5168	C++/C# Software Developer
5324	C++/C# Software Developer
5572	C++/C# Software Developer
5846	C++/C# Software Developer
6233	C++/C# Senior Software Developer
6829	C++ Senior Software Developer
8572	C++/ C# Senior Software Developer
8795	C++ Senior Software Developer (Linux)
8796	C++ Senior Software Developer (Windows)
8797	C#/C++ Senior Software Developer
8798	C++ Software Developer (Windows)
8799	C#/C++ Software Developer
8800	C++ Software Developer (Linux)
9171	C++ Software Developer (Linux)
9291	C++ Software Developer (Linux)
9977	C++ Senior Software Developer
10603	Senior C++ Developer (Unix)
17615	C++ Software Developer
17621	Senior C++ Software Developer

Figure 2

2.2 Keyword Extraction

In order to extract keywords from text, we first create terms in Unigram and Bigram for each text. We create directed graph based on sentence structure. Each term is represented by each nodes. There are in-coming edges from previous terms, and out-coming edges towards next terms. Then we collect term features including in degree, out degree, total degree, term frequency, average location score which represents the location of the term in sentences, TFIDF score and headline score which is the frequency of terms in titles. We used HITS ranking algorithm on directed graph to find important nodes. We rank terms based on collected features and HITS results and then retrieve key words based on the rank of terms. In each job cluster, majority vote is used to on each jobs' keyword list to retrieve keyword lists as representative for the entire cluster.

Figure 3 shows the result of keyword extraction from a job requirement. Figure 4 shows the result of keyword extraction from a course description.

- Bachelors degree in Computer Sciences or a related discipline;
- Over 3 years of Object Oriented C++ development, enterprise-class system architecture and design or equivalent combination of education, skills and experience;
- Extensive knowledge of UNIX platform technologies including threading and sockets;
- Demonstrated record of designing and implementing high quality software products delivered to market;
- Real time programming experience;
- Strong problem-solving skills and ability to be a successful member of a team;
- Good English language skills and ability to develop those skills;
- Desire to learn new technologies and in future move to .NET platform;
- Knowledge and application of software development methodology (preferably UML).



object oriented c++ development
good english language skills
real time programming experience
unix platform technologies
software development methodology

Figure 3

' Understand basic mechanisms of the OOP paradigm : classes, interfaces, inheritance, polymorphism, ; etc
. Develop programs with conditionals and loops Design and implement recursive algorithms Writing, compilin
g, and running basic Java applications Introduction to Android : Installation and setup of Android SDK and Androi
d emulator '



basic java applications
android emulator
android sdk
recursive algorithms
oop paradigm
basic mechanisms

Figure 4

2.3 Keyword Lists Matching

Now we have a list of key words for each job cluster, the next step is to match job clusters and courses based on extracted key words. TF-IDF values are used to vectorize keyword lists both from job requirements of job clusters and course descriptions. The value in each vector is TF-IDF value for corresponding words. Cosine similarity is used to measure the closeness of two vectors. We used cosine similarity to rank the similarity of vectors created from key words lists in job cluster and course description. For each job cluster, courses are ranked by the relativity of course content and job cluster requirements.

3. USER INPUT MATCH

When user type in search words in the search bar of our website. We calculate Jaro Winkler Distance Score between the typed search words and existing job titles. The value of Jaro Winkler Distance Score is between 0 and 1 where 1 means exactly the same and 0 means no similarity at all. Figure 5 shows an example of how Jaro Winkler Distance reflect the job title similarity to users' input. Clearly, Software Developer is much similar to Software engineer than Accountant is.

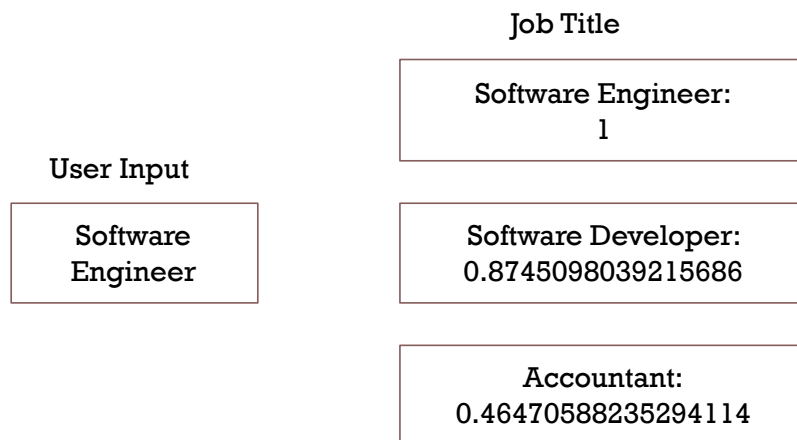


Figure 5

IV. USER INTERACTION

We used Django Framework to build our website. There are two advantages of using this framework. One is its support of Model View Controller programming paradigm which allows us to keep user interface and logic layers separated. This helps us to simplify and speed up development. The second advantage is its python language which makes us convenient to access python libraries for text processing.

Figure 6 gives an example of a use case. When user inputs ideal job which is “data scientist” in the search bar, list of courses that matches “data scientist” requirements shows up on the right. Once user click on the courses they found might be interest, they will be redirected to the main page of courses in corresponding platforms.

Many other existing course search engine are keyword based. If user search courses with job title, courses that are important to job requirement but not containing key words in job title could be easily filtered. For example, when user types in “data scientist”, courses contain key word “data scientist” are ranked high. However, “data scientist” jobs would require candidate have knowledge in NLP(Natural Language Processing). NLP related courses could be filtered out by key word based search engine if they don’t have contain key words “data scientist” in content. As shown in Figure 7, We used Class Central, one of the

most popular online course search engine, to search “data scientist”. However, we didn’t get any NLP related course because many NLP courses don’t contain key word “data scientist”. Our search engine fixed this problem since its search is based on job requirements.

The image shows two screenshots. The top screenshot is from 'MOOC 2.1 Course Finder'. It has a header with 'MOOC 2.1 Course Finder', 'Login', and 'Register'. On the left, a sidebar titled 'I want to be...' contains a text input with 'data scientist' and a blue 'search' button. The main area displays a list of course results, each in a box with the course title and provider: 'Hands-on Text Mining and Analytics by coursera', 'Text Mining and Analytics by edX', 'Applied Text Mining in Python by coursera', 'Natural Language Processing (NLP) by edX', 'Natural Language Processing and Language Understanding in Educational Research by edX', 'Natural Language Processing by udacity', 'Sequence Models for Time Series and Natural Language Processing by coursera', 'Visualizing Data with Python by edX', 'Data Science Research Methods: Python Edition by edX', and 'Data Science: Capstone by edX'. The bottom screenshot is from the Coursera website. It shows the course 'Hands-on Text Mining and Analytics' by Min Song, offered by Yonsei University. The course has a 4.0 rating from 38 reviews. A button says 'Enroll for Free Starts Apr 20' with a note 'Financial aid available'. It also states '10,503 already enrolled'. The top navigation bar includes the Coursera logo, an 'Explore' dropdown, a search bar with the placeholder 'What do you want to learn?', and links for 'For Enterprise', 'Log In', and 'Join for Free'.

Figure 6



























 Data Analysis with R Udacity  / Syllabus	Self paced	 18 Reviews
 Intro to Data Science Udacity  / Syllabus	Self paced	 13 Reviews
University of California, Davis  SQL for Data Science Coursera  3-5 hours a week , 4 weeks long  / Syllabus	27th Apr, 2020	 1 Reviews
Johns Hopkins University  The Data Scientist's Toolbox Coursera  1-4 hours a week , 4 weeks long  / Trailer / Syllabus	20th Apr, 2020	 162 Reviews
IBM  Databases and SQL for Data Science Coursera  2-4 hours a week , 4 weeks long  / Syllabus	20th Apr, 2020	 1 Reviews
IBM  Data Visualization with Python Coursera  4-5 hours a week , 3 weeks long  / Syllabus	20th Apr, 2020	 1 Reviews
University of California, San Diego  Probability and Statistics in Data Science using Python edX  10-12 hours a week , 10 weeks long 	28th Apr, 2020	 11 Reviews

Figure 7

V. COLLABORATION

We collaborated with Team MOOCer 2.0 to retrieve course information from online courses platforms. We are responsible for courses in edX and they are responsible for courses in Udacity and Coursera. Since their focus is profile management, our job skills based search could provide their user extra search option when searching for courses.

VI. FUTURE WORK

The quality and quantity of data collected from online sources are key factors to the success of our matching model. For clusters don't have enough jobs in it, the matched courses could be biased. Therefore, we could adding more jobs and course sources to create more representative job clusters and improve our matching models.

The users' behavior could also reflect the usefulness of certain courses. After we have enough users using our website, we could also add popularities and ratings

of courses matched to certain key words as feature to improve our matching model.

VII. Conclusion

In this project, we developed website in Django framework for users to search online courses across multiple platforms with job key words. A matching model is built by clustering similar jobs, extracting key words from job cluster requirements and courses descriptions and matching job clusters and courses based on key word lists. More job and course data and features such as user feedback could be introduced in building model to further improve the course matching accuracy and reduce bias.

Reference:

1. Pennington, J. (n.d.). GloVe: Global Vectors for Word Representation. Retrieved March 10, 2020, from <https://nlp.stanford.edu/projects/glove/>
2. Lutins, E. (2017, September 15). DBSCAN: What is it? When to Use it? How to use it. Retrieved March 10, 2020, from <https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818>
3. Litvak, M., & Last, M. (n.d.). Graph-Based Keyword Extraction for Single-Document Summarization. Retrieved April 2, 2020, from <https://www.aclweb.org/anthology/W08-1404.pdf>
4. Ma, E. (2018, July 26). 3 basic Distance Measurement in Text Mining. Retrieved March 29, 2020, from <https://towardsdatascience.com/3-basic-distance-measurement-in-text-mining-5852becff1d7>
5. Dreßle, K., & Ngom, A.-C. N. (2015, September). On the Efficient Execution of Bounded Jaro-Winkler Distances. Retrieved April 4, 2020, from https://www.researchgate.net/publication/281550860_On_the_Efficient_Execution_of_Bounded_Jaro-Winkler_Distances