# CS-GY 6513 Project Proposal:

# Resume matching using text processing techniques

***Elastic Computing Infrastructure sponsored and underwritten by IBM Power Systems Academic Initiative***

|  |  |
| --- | --- |
| Student name: Shuyang Cao  New York University  Brooklyn, NY  mail\_id: sc7813 | Student name: Jianfei Zhao  New York University  Brooklyn, NY  mail\_id: jz3766 |

1. **Introduction**

In recent years, text processing techniques has become very useful to solve some practical problems. For example, text classification, text searching, sentiment analysis, recommending system etc. After transferring texts into vector space, we can use multiple techniques to find some properties of the texts or quickly searching some crucial information of the given requests. These techniques include word embedding, N-Gram, TF-IDF or base on some machine learning model such as SVM, K-Means, Random Forests etc.

In this project, we aim to match a given resume to some job information which were crawled from a job hunting website. This will help the job-hunters to find some jobs which is most related to their skills, background and experience. We also plan to do some analysis on the data that collected from the website, for example, what kind of techniques are needed most in a job? The location distribution of job demanding etc. We will visualize the results and find some interesting patterns of these data.

1. **Method**

The data for the project is gathered from a job hunting website http://www.monster.com. This website provides us with tons of job information by querying the job and preferred working location. In the project, we collect the information of 20 kinds of job. By searching one job we can get tens of thousands of records. Gather detailed information and store them into .csv file and MySQL in Azure.

Crawling in python use modules of requests, lxml and multi-threading. The website of Monster is using Ajax. Thus we simulate the GET response with the headers. There will be 25 records in one page. Traversing all the pages it come out can get a list of job detail links. Access all the urls and enter second-level page. Using xpath to match the elements such as job title, company name, working location, detailed description etc. Clear the data. And store them in different tables named by the jobs in an Azure MySQL database. Write a client to query the table base on location, job, company, description.

After gathering and storing the data, we can use some text processing techniques to finish our tasks.

Firstly, we use N-Gram to extract N consecutive words in a given corpus. The concept of N-Gram is based on an assumption: the occurrence of the  word is related to previous n-1 words, and is not related to the other words. After using N-Gram, the output is a sequence of N items. The crucial part of N-Gram is to extract more features of a given text, specifically, to find a feature combination of N adjacent keywords.

Secondly, we analyse word and document frequency using TF-IDF. TF stands for term frequency, that is how frequently a word occurs in a document. There are words in a document, however, that occur many times but may not be important. in English, these are probably words like “the”, “is”, “of”, and so forth. We might take the approach of adding words like these to a list of stop words and removing them before analysis, but it is possible that some of words might be more important in some documents than others. To distinguish these words, we use another method called inverse document frequency (IDF), which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents.

In practical, we define a new vector space representation. For document i, we construct a vector such that the j-th coordinate is given by:



The term represents term-frequency and counts the number of occurrences of word j in document i. The term idf(j) represents inverse-document frequency, and is defined as follows. Let be the number of documents in the database which contain at least one occurrence of word j. Then,



Finally, we calculate the cosine-similarity between query vector and the TF-IDF matrix which is defined above.