Resume Analyzer for job descriptions using NLP

Prithika Vijayakumar
prithika@knights.ucf.edu

Priya Sudharsanan
priya.sudharsanan@knights.ucf.edu

Priyanka Siripurapu
priyankasiripurapu@knights.ucf.edu

Argula Deeksha
arguladeeksha@knights.ucf.edu

1. MOTIVATION & PROBLEM STATEMENT

Most job seekers apply for jobs without even reading the job description, this adds on to too many unsuitable resumes being analyzed which is time consuming and there are also situations where job seekers are employed in job roles that totally differ from their acquired skill set. So, the candidates will be unsatisfied with the job. For example, if the person is a good Java developer and if the company hired him/her for python opening and made to work on the skill that he/she is not good at or interested in makes him/her unsatisfied at work.

Moreover, if there is a vacant position in the company, the company will need a best possible candidate for the vacancy. The process of analyzing and extracting the sections like work experience, skill set etc, from large corpus of data(resumes) manually executing is time consuming and tedious. So, it is clear that this way of resume analysing is not efficient, flexible and also time consuming.

Also, while applying for a job , candidates are expected to upload their resumes in particular format, if the applied resumes are not in the expected format it will be difficult for analysing them, so there is need for a system that analyse the resumes in any format. Hence, we are designing an automated system that extracts information from several unstructured resumes of different formats and convert them to structured data using Natural Language Processing techniques, that is required for job matching. The developed model will save the time of the candidate as they can upload the resume in any format and also as the model analyse the resume and extracts the skills of the candidate and compare it with the job description, he/she will be placed in the job that best suites there skills and be satisfied. So, the developed model will be helpful for the company by choosing the best possible candidate for the job opening and also for the candidate by getting placed in the job that matches best for his/her abilities and skills.

Building an automated system in order to extract information from unstructured resumes and converting it into structured format. Also ranking them based on the information extracted and job description by the company using Natural Language Processing techniques.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. RELATED WORK

Most of the projects use algorithms for classifying the attributes like skills, education and experience to their specific categories or labels. The input in this case will be few resumes and their corresponding categories as training data. The output will be the prediction of suitable job when test data is fed.

In one of the approach, Name Entity Recognition is used for summarising the resumes, by extracting the attributes from the resume like name of the institute that candidate studied, education background, skills. It is common that the resumes have excess information that is irrelevant to what the recruiter is looking. Also the process of evaluating bulk resumes is time consuming and hectic task. So, with the Named Entity Recognition model the resumes can be evaluated by just looking at the important information of the candidate which reduces the time and effort required to shortlist the candidate from the bulk applications.

But, this method only summarises the resume but not gives the information if the particular resume is suitable or not suitable for the given job description. So, the resumes need to be further looked by the recruiter which is time consuming.

There are few projects which have different implementation techniques for scoring the Resume.

1. One approach for ranking the resume is done by training the model with the set of resumes that are prescreened by the experts and are classified into suitable(y=1) or unsuitable(y=0). Here ranking a resume is considered as the classification problem, which is solved by the logistic regression algorithm. Only the important features like education, skills etc, are considered leaving behind the features that are less important like notice period, expected compensation etc to make the model simple. Initially, consider X as a vector which consists of features such as experience, education, skills etc. All these features decide which resume is suitable and which resume is not suitable.This gives the information only about the particular resume being analysed, that is if the resume is suitable for the job or not but this doesn't rank the resumes.

2. Another approach that we came across for ranking the resume based on domain like finance, engineering, management etc. it extracts the semi-structured text format in a CV and the ranking is done based on keywords used in the resume. Resume is analysed and probability of the resume to fit for different domains is determined. To achieve this, they have divided it into 3 basic segments. First segment consists of segmenting the entire CV by converting it to HTML, the second segment is extracting the structured form of data from the unstructured data and the third segment is finding the probabilities of resume for different domains. Initially system is trained with few keywords. Named Entity Recognition, keyword Match is used in these kind of systems. Using these models the resumes can be rated to which domain will that be most likely be fitted.

3. OVERVIEW

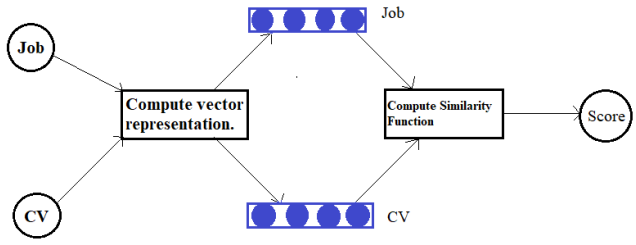
Our proposed project provides an list of best matched resumes for a given job description by analysing the entire database of resumes that were submitted by applicants. It picks a set of resumes over all resumes submitted that best matches a given job description.

To attain such results the proposed system assigns a score to each resume by intelligently comparing them with the given job description. This way we obtain a fraction of the original number of resumes of the applicants that best fit compared to the others. This decreases the number of resumes to be analysed for future selections.

Our Project will consist of 4 major operations

- Data collection
- Model training
- Extracting sections
- Scoring

Figure 1: Resume Analyser



In this project mainly three data sets are used:

- Job descriptions collected from Kaggle
- Resumes/CV's collected from various sources like Indeed etc.
- StackOverflow data dump to train Word2Vec model.

Job descriptions collected from kaggle may have unnecessary information incorporated in it which is not helpful for ranking the resumes, such information is filtered out from the description and cleaned job descriptions are obtained.

Resumes are collected from the sites using python code. The collected resumes may have a lot of information which is not required for ranking them. Sometimes candidates provide too much information that is irrelevant and not considered by the recruiter for considering the candidate for the job such irrelevant information is removed from the resumes for analysing them further.

Model is trained using corpus from stackOverflow, data collected from there is in .xml format which will be easy for training the model, initially sentences are extracted from the corpus, later words are extracted from these sentences and words are used as as input to word2vec model. Gensim implementation of word2vec is used to train the model.

Word2Vec is as shallow network which is used to learn words representations,it is based on the concept that words that are used in the similar context share similar meaning and also share similar vector representation in the vector space, word embeddings which capture both meaning and context of the word are given to word2vec as input. For example, cat, kitten are often used in similar context and also referred as cute many times and so all these words have and share similar vector representations.

From the above example we can say that all the words that are used in similar context share same vector representation and words that are related are placed together in the vector space. All the unique words in the corpus are represented as vectors in vector space such that all the words that share same meaning and used in similar context are placed in close proximity in the vector space.Predefined models in python from spacy and gensim which are trained with google news data can be used but these will not be useful to distinguish the context of technical words used in the project/resumes. For example, words like ruby and HTML will found have higher similarity when trained using predefined models than using the developed model. So,dataset that has sufficient technical words also non technical words should be used to train the model, stackOverflow data is used to train the model.

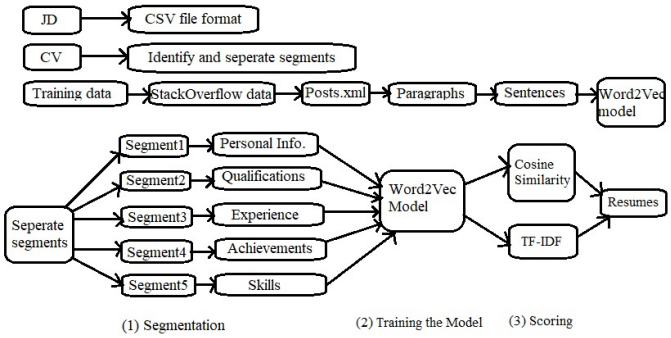
In the job description all the stop words are removed and only few important words are selected and these words are used as the keywords, for all the keywords similar words are identified and similarity is calculated. tf-idf is calculated for each word, to find if the word is informative or not. Finally, Score for CV's are calculated as the product of tf-idf and similarity.

4. METHODOLOGY

We provide a detailed description on how we would implement each module stated above.

1. **Data Collection:**
   In the data collection section, the source data required

Figure 2: Resume Analyser



for the model is collected and used for the implementation of our proposed project. These data are obtained from several publicly available datasets to train our model. We use three types of datasets:

- Resume Dataset
- Job Description Dataset
- Post from StackExchange as corpus

where posts from stackexchange will be used for training the word2vec model, the next job description dataset will be used to test the model and the resume dataset will be used to validate the model.
Resume dataset is obtained from different sites with features like name,experience, skills and education. Resumes have the technical information of the candidate who is seeking/applying for the job. Basic information of the candidate like name, email address will be present in every resume in common. Also, education details like history of his/her education like degree and institution from where degree has been obtained is mentioned in resume.Past working history and work experience is mentioned in few resumes. Achievements of the candidate may be included in the resume along with extra activities like hobbies.These resumes may have different formats or may not have certain structure. We have taken a sample of 22 resumes. It will be difficult to extract the resumes in bulk from different sites.So, we have implemented python code to extract the resumes from these sites. All the resumes that are copied to the clipboard is saved in to mentioned folder while the python code is running.Thus, instead investing manual effort to extract resumes from sites, python code is used to automate this process of resumes extraction.
Job descriptions dataset is also obtained from kaggle with features like :

- Title
- Description

Job description have the information that is expected from the job applicants.Few examples for titles may be as follows :

- Software developer
- Product owner
- Project Manager
- Technical Lead
- Data Analyst
- Senior Systems Engineer
- Web developer
- Systems Engineer
- Technology Analyst etc.,

These Job descriptions are posted by companies expecting the candidates for a particular role as mentioned in the title by having the requirements in his/her resume as that described in the job description.
We are using stackexchange dump to train our word2vec model as it will be more efficient in identifying the technical differences. This dump is user contributed content on the Stack exchange website.We got this dump from **https://archive.org/download/stackexchange**. The data in this dump is in xml format, the root source for this dump is from anonymized people. We have extracted 66 folders from stackexchange directory for training our model.Different site consists of data in xml format placed in compressed .7z files.Python code is used to extract the files from the compressed zip.Each compressed files will have .xml files like :

- PostHistory.xml
- Posts.xml
- Votes.xml
- Users.xml
- PostLinks.xml
- Comments.xml

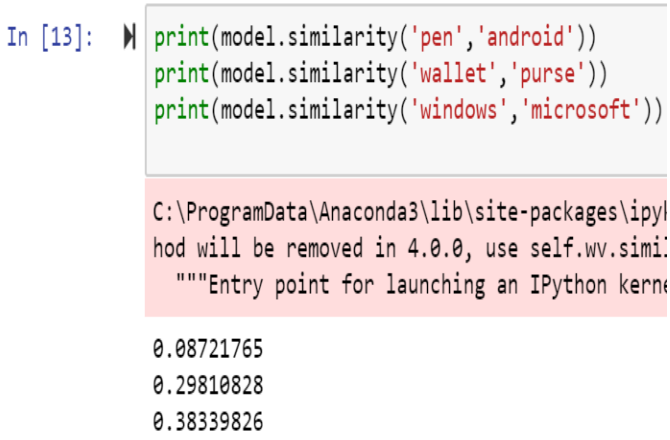
We are training our word2vec with the stackexchange dump using gensim library in python. Gensim is used for retrieving similarity and topic modelling. All the words if the stackexchange dump/corpus is represented in the vector space.
Data present in posts.xml is used to train the model, following information is present in posts.xml:

- Id
- AnswerCount
- ParentID (It is present only if PostTypeId is 2)
- LastEditDate = ”2008-04-06T20:28:34.823”
- FavoriteCount
- CreationDate
- Body
- ViewCount
- ClosedDate = ”2008-02-10T12:51:01.480”
- OwnerUserId
- LastEditorDisplayName
- Score
- LastEditorUserId
- PostTypeId a)Question b)Answer
- LastActivityDate = ”2008-02-10T11:51:01.480”
- Title

Figure 3: stackexchange dump

Name	Last modified	Size
Go to parent directory		
3dprinting.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:28	548.6K
3dprinting.stackexchange.com.7z (View Contents)	02-Mar-2020 13:29	9.6M
Sites.xml	02-Mar-2020 15:55	337.1K
academia.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:29	4.2M
academia.stackexchange.com.7z (View Contents)	02-Mar-2020 13:30	114.4M
ai.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:30	767.4K
ai.stackexchange.com.7z (View Contents)	02-Mar-2020 13:30	16.6M
android.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:30	2.8M
android.stackexchange.com.7z (View Contents)	02-Mar-2020 13:31	96.4M
anime.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:31	3.7M
anime.stackexchange.com.7z (View Contents)	02-Mar-2020 13:31	28.3M
apple.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:31	4.0M
apple.stackexchange.com.7z (View Contents)	02-Mar-2020 13:33	205.8M
arduino.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:33	870.6K
arduino.stackexchange.com.7z (View Contents)	02-Mar-2020 13:33	55.3M
askubuntu.com.7z (View Contents)	02-Mar-2020 13:37	750.2M
astronomy.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:37	683.8K
astronomy.stackexchange.com.7z (View Contents)	02-Mar-2020 13:37	25.1M
aviation.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:37	2.0M
aviation.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	65.6M
avp.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	519.0K
avp.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	14.2M
beer.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	232.8K
beer.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	3.1M
bicycles.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:38	1.4M
bicycles.stackexchange.com.7z (View Contents)	02-Mar-2020 13:39	45.2M
bioinformatics.meta.stackexchange.com.7z (View Contents)	02-Mar-2020 13:39	267.5K
bioinformatics.stackexchange.com.7z (View Contents)	02-Mar-2020 13:39	7.3M

Figure 4: Word2Vec model



```
In [13]: print(model.similarity('pen','android'))
         print(model.similarity('wallet','purse'))
         print(model.similarity('windows','microsoft'))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipyk
hod will be removed in 4.0.0, use self.wv.simi
  """Entry point for launching an IPython kerne
```

```
0.08721765
0.29810828
0.38339826
```

- Tags
- CommunityOwnedDate = "2008-02-10T12:51:01.480"
- AcceptedAnswerId(It is present only if PostTypeId is 1)
- CommentCount

As all information present in the dump as .xml format it is easy to access the data from .xml files and use as input to the training model.
Using Resume data set and Job Description dataset,we are training a word2vec model that represents words of the respective datasets in vector space are those are compared using cosine similarity to find the best match. The resume datasets are collected from the kaggle and fed to the model. We extract the skills and experience from each resume of the resume dataset and compare them to the job description. The output will be the predict if the resume is suitable for the given job description.

2. Model Training
We use word2vec model to train the datasets. Our propsed system will implement the Word2Vec word embedding technique which creates word vectors with Python's Gensim library.
The Wor2vec approach uses deep learning and neural network approach to transform words into corresponding word embedding represented as vectors in the N-dimensional vector space such that semantically similar words appear close together. Words that share common contexts in the corpus are placed closer to one another in the vector space. To implement word2vec we require a corpus of data that is obtained using data extraction which is pre-processed and feed into the model. The pre-processing includes extracting sentences from the resume using a python library called BeautifulSoup. Now these sentences are fed into the Word2vec model to train the model. Using the genism implementation of Wor2vec, our model will be trained and these produces a vector array for each word.
The following command is used to train our model: model = gensim.models.Word2Vec(sentences, workers=4, size=300, min_count = 1, window = 15, sample = 1e-3),where we have set the size (Word vector dimensionality) to be 400, workers to indicate number of threads to run in parallel,min_count represents the minimum word count and context window size is set.

We are testing the model by giving some set of data and finding the odd one out to check the performance of our model. Similarity score is printed for some pair of words to determine its accuracy.

3. Data Extraction and Sections extraction from resume

1. Each file is extracted from the stack exchange and placed in directory folders.
2 . All the resumes collected from kaggle site with different formats like .pdf and .docx are converted into .txt format
3. In job description.csv file all stop words/unwanted words are removed the remaining words are lemmatized.
Below steps are followed to extract the sections from the resume :
1. Dictionary with the sections like 'education' , 'experience' , 'skills' and 'miscellaneous' as the keys and related categorises under each section as corresponding pairs is considered.
2. Each resume is parsed line by line and during parsing empty lines, lines starting with the symbols are removed and the words from the remaining set of lines are lemmatized.
3. Each line is categorised into one of the sections in the dictionary by considering its similarity with all the sections and placed into the section that holds the highest match/similarity.
Below is the output screenshot of the resume data in sections like :

- education
- experience

Figure 5: Section Extraction

Out[17]:

	cv2	cv3	cv36	cv4	cv50	cv51	cv52
edu	education \n B. Tech E.C.E \n Brilliant Colleg...	education \n MCA Computer Science \n Patna Sci...	education \n B.E CEG \n Anna University Chenna...	education \n bsc \n Mumbai university \n	education \n Bachelor Engineering Electrical E...	education \n B.E. Computer Engineering \n MIT ...	education \n MSc Computer Science \n Pune Univ...
exp	work experience \n Fresher \n MAJOR PROJECT - ...	dream have career help enhance skill set , hel...	work experience \n application developer \n IB...	work experience \n Fresher \n description ...	â□¢ experience work Tomcat Web Server . \n â□¢...	work firm professional work drive environment ...	work experience \n Fresher \n significant cont...
skill	seek position utilize skill ability	skill \n core Java (1 year) ,	Tools & Technologies MDM Collaborative		â□¢ 5 year software development experience	iNautix Technologies - Pune , Maharashtra	technology : APPIAN \n role : application

- skill

4. Scoring
   We will be using pretrained Word2vec model to generate word vectors for the words in the resume and job description.Hence the first step in scoring would be to load the Word2vec model.
   The extracted resumes (section extraction) is our input for scoring.
   To score a given resume with respect to a job description we first pre-process (includes removing stop words , lemmatization)the job description to get a set of keywords (features) that we will be looking into the resumes.
   For each of these keywords we find similar words and calculate the TF-IDF for the same.TF-IDF stands for term frequency-inverse document frequency assigns some weight to the word based on the number of occurrences in the document also considering the frequency of the word in all the documents.
   using the TF-IDF obtained above we would assign a score to the resumes using the TF-IDF obtained and the similar words that were generated in the above steps.
   The tf-idf (Term Frequency-Inverse Document Frequency) is basically used to weight words according to how important they are and how frequently they are used in the documents. And all the frequently occurred words are given a higher weightage. Once we obtain the tf-idf for each word, then we score the CV by considering both tf-idf value and the similarity value. The score can be calculated by tf-idf * similarity for all the generated words.The above scoring is just a proposed plan, if we do not obtain accurate results we would utilize an alternate scoring function to provide better results.

5. EVALUATION

We perform evaluation by scoring every resume that fit the best for the provided job description (JD). To score a particular resume we calculate TF-IDF value along with Cosine similarity. We will remove all the stopwords, symbols etc., for a given Job Description. Then words that are in the resume are mapped to the sections such as education, work experience skills etc. And for each keyword that is found we will find five similar words and their corresponding similarity. Then we have found the TF-IDF value for the previously obtained words. TF-IDF is the Term Frequency - Inverse Document Frequency that is used to summarize the documents. A Term Frequency is used to count how many times a word has occurred in a given document. The Inverse Document Frequency (IDF) is the number of times a word appears in a collection of documents. The TF-IDF value depends on one factor which is the weightage. For example the words which are used frequently in many documents will have a lower weightage while the other words that appear infrequent will have a higher weightage. So, words such as this, what, if will rank low because they are common in all the documents and they don't mean much to that particular document. The TF-IDF value is calculated by multiplying two important metrics: how many times that particular word has appeared in a document, and the inverse document frequency of the word across a collection of documents. Mathematically the TF-IDF score for a particular word t in the document d from the document set D is calculated as follows:

$$tfidf(t, d, D) = tf(t, d).idf(t, D)$$

where:

$$tf(t, d) = log(1 + freq(t, d))$$

and

$$idf(t, D) = log(\frac{N}{count(d \in D : t \in d)})$$

The score of the CV is the sum of (tf-idf * similarity) for all the generated words. Here we have used the Cosine similarity to score the CV. The Cosine similarity is used to measure how similar the words are or how close they are to each other in terms of their context or meaning. Cosine similarity is also used to find the text similarity between two documents irrespective of their size. So, here a word will be represented into a vector form. Mathematically, Cosine similarity will measure the angle between any two vectors that are projected in a multi-dimensional space. And the Cosine similarity of two documents will be in a range of 0 to 1, where if the similarity is 1 then it means that the two vectors have the same orientation. Cosine similarity is given by:

$$Similarity = cos(\theta) = \frac{A.B}{||A||||B||}$$

$$= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

In our model using the above formula we have calculated the cosine similarity for two vectors: Software developer and Web developer. And we have got 64 percent of similarity between those two vectors. We will calculate the similarity for all the other words until we get the closest word to the given vector. The one with the highest similarity is indicated as 1 which means that the two vectors are same. And that CV is given a highest score and with this process of scoring it becomes easier for the Human Resource manager to pick the best fit CV for the given job description without any

Figure 6: Ranking of resumes

```
In [28]:
ranked_resumes = []
for i in range(num_resumes):
    ranked_resumes.append((score[i], i))
ranked_resumes.sort(reverse = True)
for a, i in ranked_resumes:
    if list(resumes)[i] != '.DS_Store':
        print(list(resumes)[i], ':', a)
```

```
cv50 : 848.2244209343917
cv61 : 290.0897429600622
cv36 : 215.21116481910263
cv55 : 94.09680660970866
cv57 : 84.65091181927765
cv63 : 84.18687413803237
cv62 : 81.41001888702947
cv56 : 77.53724306621127
cv66 : 74.32732285996798
cv67 : 71.85267783385399
cv59 : 63.895706645456286
cv2 : 55.724128414942214
cv64 : 41.76674053098422
cv65 : 39.777480352150306
cv3 : 36.41633911877955
cv54 : 36.0104465624718
cv58 : 35.42511191790185
cv53 : 30.781095244701476
cv52 : 23.874102696449867
cv51 : 17.862940586667772
```

further manual selection.
Below is the output screenshot of resumes scored according to the best match with job description.

There are other similarity metrics like Euclidean distance, but we have not considered it because if the two documents are far apart by Euclidean distance, there are still chances that they are close to each other in terms of their context.

6. EXPECTED OUTCOMES AND RISK MANAGEMENT

The expected outcome is the reduction in the number of resumes to be analyzed for a given job description. And this is done by finding words from the resume that are similar to the keywords in the job description and assigning a score to each resume using TF-IDF and Cosine similarity. So, the TF-IDF will calculate the frequency of word in the document. So from all the resumes, we are reducing the number by finding similarity with the job description which will make the work a lot easier. And the Cosine similarity is used to to measure how similar the words are with each other.

Code Execution:
The order in which the code was executed is given below:

1. The extracted dataset is stored in stackexchange folder.
2. parah-checkpoint.ipynb is then executed to generate paras file for the posts.xml file in each dataset
3. sentence-checkpoint is then executed to generate sentences file for each para file created in previous step.
4. model-checkpoint.ipynb is then executed to train the word2vec model using the sentences files obtained.
5. This trained word2vec model is saved as Word2vec-model in our project.
6. Next the collectCV.py code inside the data folder is executed to copy the CVs from clipboard and executing the code results in a separate CV folder which stores the resumes.
7. The job description is stored as jd.csv file
8. SectionExtraction.ipynb file is executed which takes the CVs as input and generates a resume.csv file
9. Finally scoring.ipynb is executed which ranks the resumes based on the job description.

Risks associated are:
In word2vec model, we can't increase the vocabulary after training. So it would be difficult to include some new keywords related to a different job description.
There are several other models which give better accuracy with complex implementation. We are trying to obtain better accuracy with simpler implementation using word2vec model.

7. PLAN AND ROLES OF COLLABORATORS

- Resumes and job description extraction from different sites like Indeed and kaggle respectively (Web scraping) and Information extraction from resumes - Priyanka
- Stackoverflow data for model training and pre-processing that data into paragraphs and sentences - Priya
- Lemmatisation of JD - Deeksha
- Word2Vec model training and Scoring resumes based in trained model - Prithika.
- Report and presentation was equally shared by all four.

8. CONCLUSION

Analysing resumes and selecting the resume that best fit the job description manually is time consuming process, to make the resume selection process easy and less time consuming automatic resume analyser by which resumes are scored according to their best fit to job description is designed. Earlier there are many automatic resume analysers, one such example is resume summarizer in which resumes are summarised using NLP techniques but after summarising resumes, they need to be evaluated manually again to know if they are suitable for job. so, we can with current model where resumes are scored based on their best fit with job description.

9. DATA AND CODE LINK:

https://tinyurl.com/y86f497c

10. REFERENCES

1. http://dspace.bracu.ac.bd/xmlui/handle/10361/9480
2. https://www.kaggle.com/madhab/jobposts
3. https://stackabuse.com/implementing-word2vec-withgensim-library-in-python/

4. https://archive.org/download/stackexchange
5. https://medium.com/@everislatam/using-machine-learningto-retrieve-relevant-cvs-based-on-job-description-4f84d5440c70
6. https://arxiv.org/pdf/1812.08947.pdf
7. https://www.researchgate.net/publication/313851778_Resume_Parser_with_Natural_Language_Processing
8. http://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/9480/14101061
9. https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html
10. https://www.machinelearningplus.com/nlp/cosine-similarity/
11. https://medium.com/@Synerzip/resume-ranking-using-machine-learning-implementation-47959a4e5d8e
12. https://vinayakjoglekar.wordpress.com/2014/06/24/rankingresumes-using-machine-learning/