WORKFORCE ACQUISITION USING TEXT MINING

Problem Description -

There has been a rise in the number of applicants for each job due to reasons such as more job opportunities, emergence of new technology, and after-effects of the COVID-19 pandemic. The first step in the job application process is the screening of resumes. HR personnel find it increasingly challenging to pick resumes that match their job descriptions. Different resumes have different styles of presentation and formats of organizing data.
This project aims to help the candidates update their resumes to better fit job descriptions, and also mainly help companies speed-up their candidate screening process. The objective of this project is to match the job descriptions with the best resumes using Cosine Similarity. We would also do topic modeling to display relevant topics for both the job descriptions as well as resumes.

Dataset -

There are two datasets for this project - The Job Description Dataset(AmazonJobs.csv) and the resume dataset (UpdatedResumeDataSet.csv).
The Job description dataset has been obtained by scraping Amazon Jobs. This dataset contains 235 job descriptions and their corresponding Job IDs.
The resume dataset has been downloaded from Kaggle and this contains resumes of 963 candidates.

Methodology -

Web Scraping - Creating job description dataset

- ➢ Using Selenium as the automation tool, we scraped around 235 Job Descriptions from Amazon Jobs. The XPATH technique was used to navigate through the different jobs in all the pages.
- ➢ We scraped the Job description data containing ‘Basic Qualifications’ and ‘Preferred Qualifications’ along with the ‘Job ID’ for each job.
- ➢ This data was stored in a Python dictionary with ‘Job IDs’ as the key and ‘Qualifications’ as the value. It was then exported to a CSV file.

Data Preprocessing - Cleaning resume and job description dataframes

- ➢ Both the data sets were cleaned initially using MS Excel to remove recurring strings and text which were in foreign languages like German and Mandarin which wouldn’t help in building the models.
- ➢ Further cleaning and preprocessing was done using Python in the following order:

- ➔ Removal of Non-ASCII characters.
- ➔ Removal of punctuations and special characters.
- ➔ Removal of other useless and repetitive words like ‘education’, ‘skillset’, ‘tools’ ‘proficiency’, ‘qualifications’, etc. based on observation.
- ➔ Converted cleaned text to lowercase.

Cosine Similarity - Calculate cosine similarity between job descriptions and resumes.

- ➢ We utilized the TfidfVectorizer function from the Sci-kit learn library to convert the documents to a matrix of TF features.
- ➢ This function facilitated stop-word removal and tokenization to be used in the CosineSimilarity function.
- ➢ We created a function called “`comp_description`” which returns top 20 resumes for any job ID based on Cosine Similarity values.

Latent Dirichlet Allocation(LDA) - Topic modeling to identify topics for both datasets.

- ➢ Using gensim.simple_preprocess, we tokenized the documents.
- ➢ Stop-word, new-line,email IDs, and other unwanted characters were removed followed by Lemmatization.
- ➢ We made use of the LdaModel function from the gensim library to get the topics.

Results and Visualization -

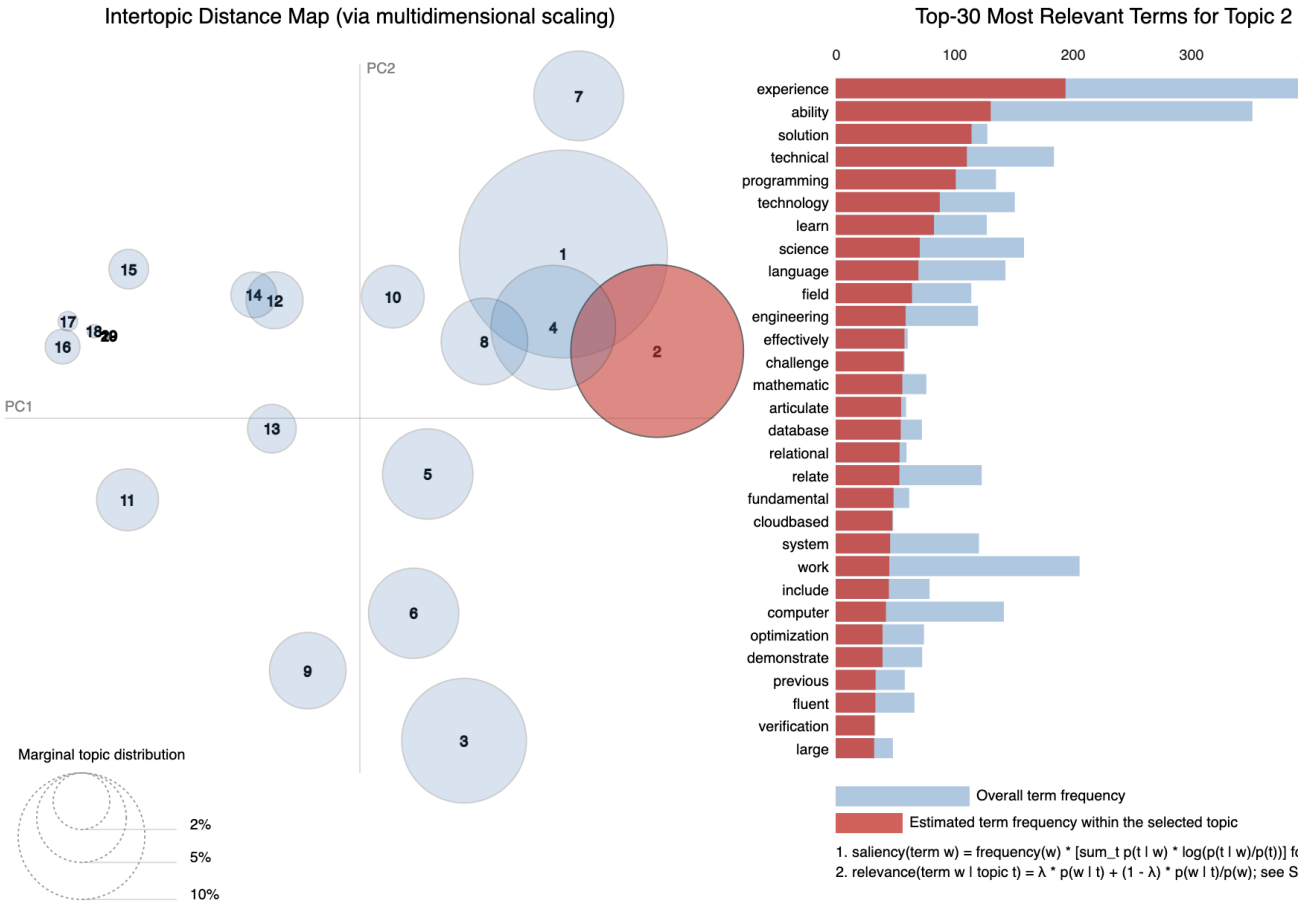
The output of our CosineSimilarity function was as shown below:

```
Complete Job Description:
---------------
  record of delivering large analytical solutions with business impact   experience on rsasmatlab and sql   excellent microsoft office
---------------
Job-ID- 200 --Resume ID :  Resume9  ---Similarity Matched= 0.6255230084786465
Job-ID- 200 --Resume ID :  Resume19  ---Similarity Matched= 0.6255230084786465
Job-ID- 200 --Resume ID :  Resume29  ---Similarity Matched= 0.6255230084786465
Job-ID- 200 --Resume ID :  Resume39  ---Similarity Matched= 0.6255230084786465
Job-ID- 200 --Resume ID :  Resume7  ---Similarity Matched= 0.5594367029135635
```

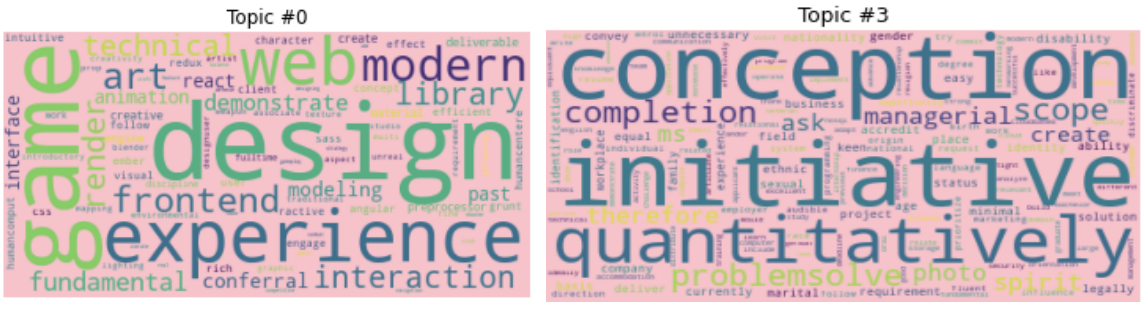
The topic clusters and word clouds obtained from the job description and resume datasets are:

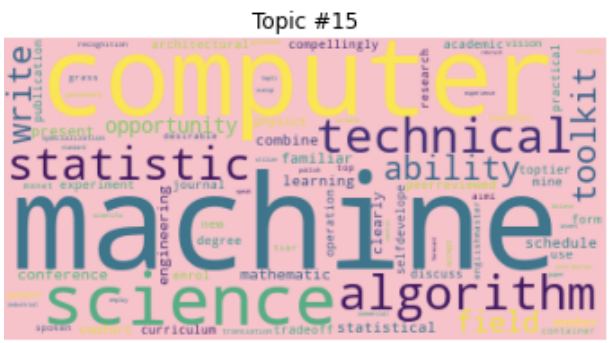
Job Description Dataset-

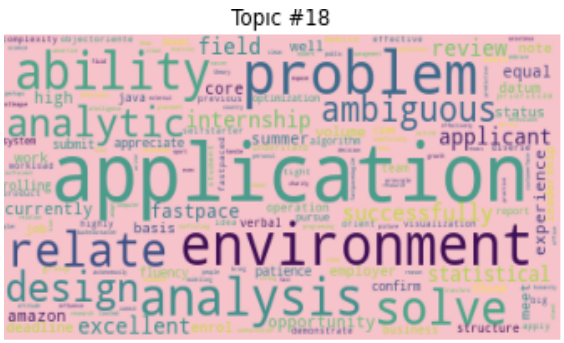
Topic Clusters using pyLDAvis library:



Wordcloud using the matplotlib and wordcloud libraries:

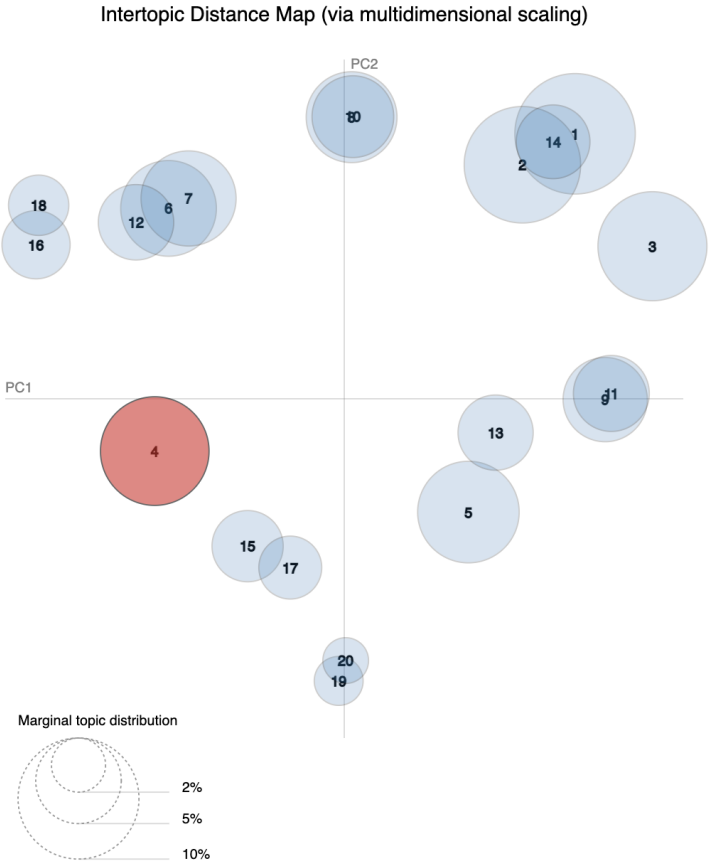


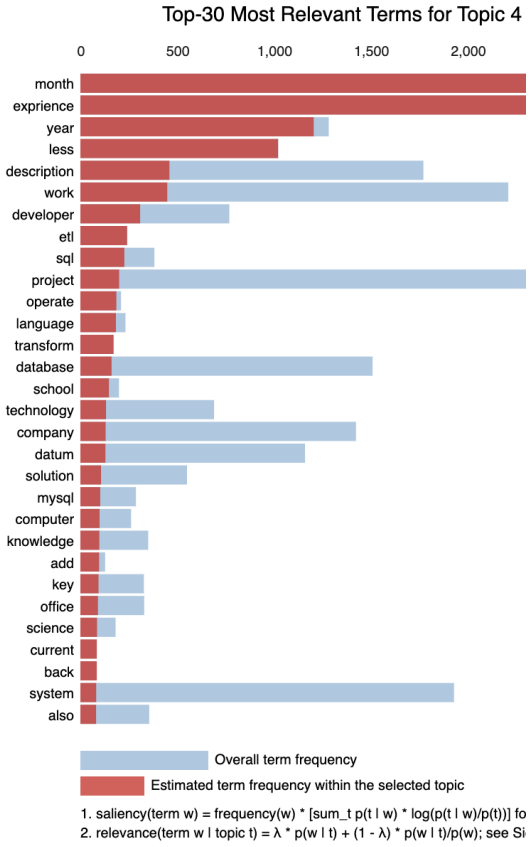




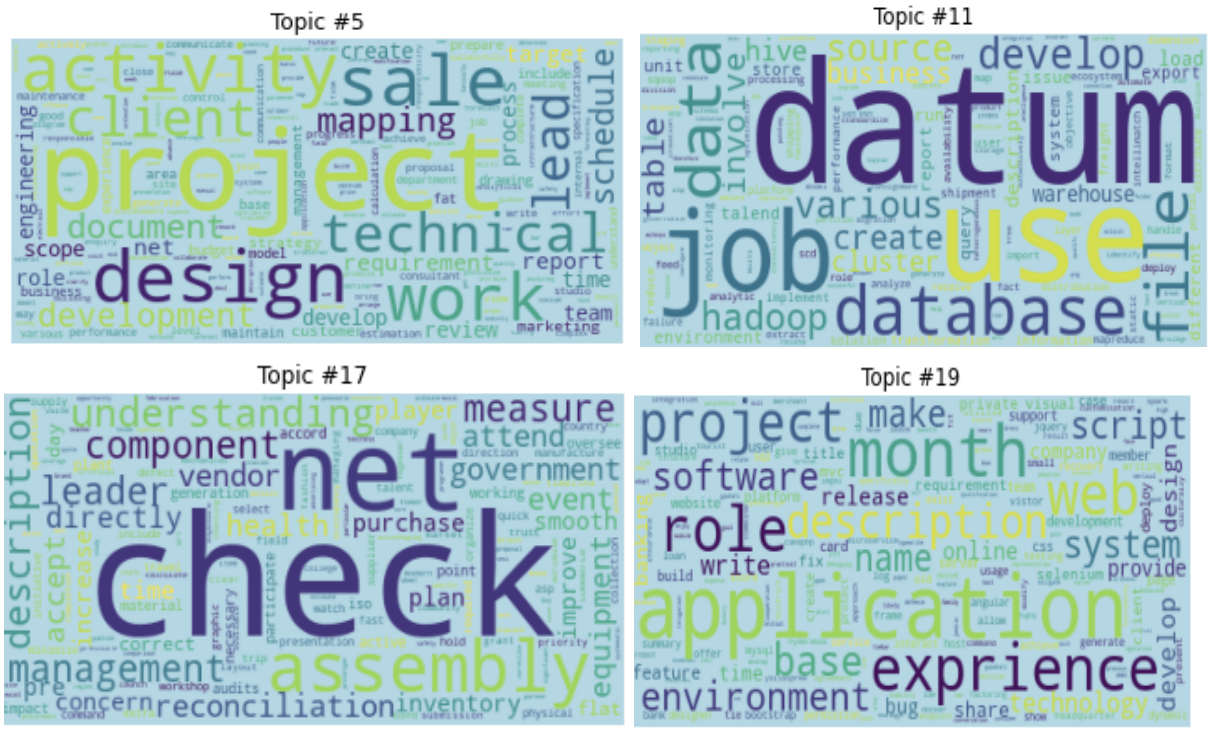
Resume Dataset-

Topic Clusters using pyLDAvis library:





Wordcloud using matplotlib and wordcloud libraries:



Coherence and Perplexity Scores: Coherence score tells us how interpretable the topics are to humans. Perplexity score tells us how well our model predicts a sample. A lower perplexity score indicates better generalization performance.
In our project, topic modeling provides the top skills present in the dominant topics which would be useful for organizations to gauge the skills available in the market.

LDA MODEL	COHERENCE SCORE	PERPLEXITY SCORE
Job Description Model	0.532	-5.772
Resume Model	0.376	-6.582

Teammate Contribution -

All the team members contributed to every aspect of the project, from research, building code, analysis and report generation, and gave it our best.

TEAMMATE NAME, UID	CONTRIBUTION
Dhananjay Singh, 668437546	100%
Shivani Narahari, 675954089	100%
Srinanda Kurapati, 663244158	100%