**Job Recommender System**

**Information Retrieval Project**

Jeffrey Aguilar, Tahura Islam, Harrown Kenta

Professor Houwei Cao

December 12th, 2022

CSCI 626 - Information Retrieval

**Table of Contents:**

1. **Abstract**
2. **References & Tools**
3. **Job Recommender**
4. **Related Works**
5. **Dataset Description**
6. **Design & Methods**
7. **Experiments & Results**
8. **Future Improvements**
9. **Conclusion**
10. **Team Contribution**
11. **References**

**Abstract:**

In the busy lives of college students looking to become working professionals in industry, looking for a good job in the vast sea of vacancies can be very time consuming. There are many job applications and finding websites such as Glassdoor, Linkedin, and Indeed where you have to manually input information to find the desired job through their search engines that are already typed into your resume. So why isn’t there a search tool where you can easily find jobs with the best match based on your resume? Well with our Job Finder Application, now there is.

**References and Tools:**

* Python Libraries
  + Sklearn
    - TFIDF Vectorizer
    - Cosine Similarity
    - K Nearest Neighbors
  + PyPDF2 - pdf reader to read resumes
  + Pandas
* GitHub - Stored Project
* Anvil - Front end Framework for Colab Notebooks
* Google Colab - Main Development

**Job Recommender:**

Our job recommendation system is a content based system which takes in application (specifically from Glassdoor jobs data for open Data Science positions). The main effect of the application is that it takes in job descriptions from this dataset and compares them against the resume of a person as a user query through TFIDF , while utilizing the SKLearn algorithms of Cosine Similarity and K Nearest Neighbor Search aid in finding similarity scores for the top 20 results.

PDFPY2 takes in Resumes and CSV Reader takes in the Jobs Dataset and Descriptions where on the colab notebook, the resumes undergo preProcessing, tokenizing, and lemmatizing. They are then vectorized and prepared to be put into the Cosine Similarity and KNN Models where they are used to be compared against one another, returning arrays of score based on similarity. The top 20 jobs for each one of these methods is printed out.

**Related Works:**

While researching for this project, according to makeen.io, we discovered that there were five job recommendation systems that are related to our own. These include:

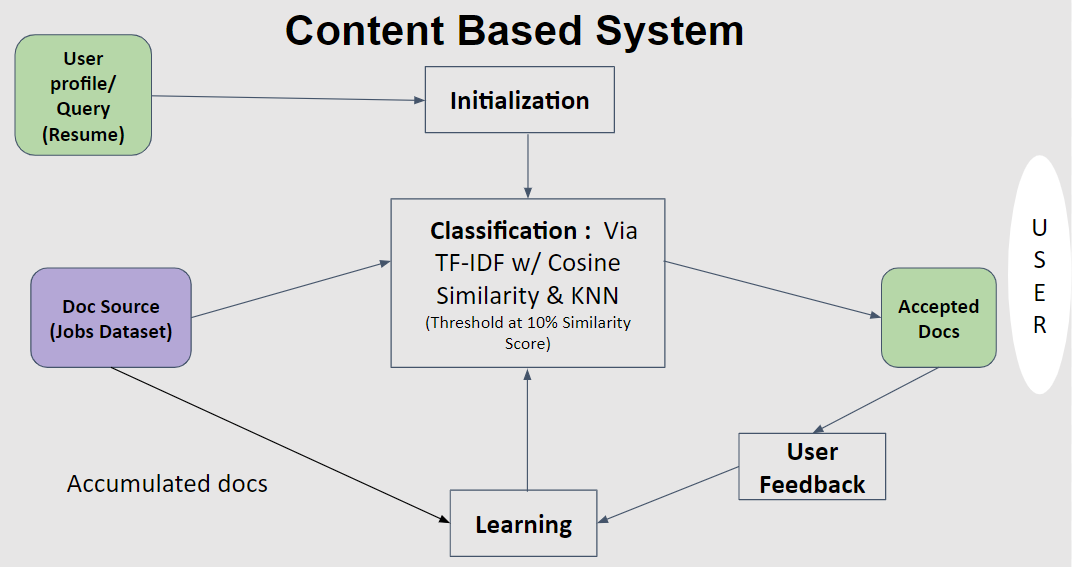
1. TF-IDF and Word2Vec Based Job Matching: This method is a combination of TF-IDF and Word2Vec. Word2Vec is a word embedding algorithm that relies on a shallow neural network to vectorize the text in a corpus. These two are performed systematically to later analyze the computed data with similarity functions. After running these two algorithms on the two sets of data, job and applicant details, the results will be ready for data matching. One of the similarity functions will be applied to the collected data to get the top similar job vectors.
2. Personalized and Localized Job Recommendations: With a set of applicants and a set of job features, the task is to match users to their preferred job features. There are two approaches to this type of matching: content-based and location-based. Both approaches have pros and cons to them, and there are also ways to combine them to take advantage of both the techniques. Word2Vec comes in handy with this method as well and plays a similar role by converting the text in both resumes and job postings such as skills, experiences, services, current city, etc. into numerical feature vectors.
3. Topic Modeling Via Matrix Factorization: NLTK based topic modeling method is designed to discover the latent semantic structure or topics within a corpus of documents, which in the case of job matching, includes both job descriptions and resumes. It is derived from the co-occurrences of words across the content of the documents. With matrix factorization, it builds topic hierarchy via splitting larger-topics into sub-topics.
4. Resume Parsing: With the process of resume parsing, all relevant data is extracted from free-form resumes. Tesseract is one of the initial-stage tools for this process. It is an optical character recognition engine that reads text from common CV PDFs or even an image. Once the text is extracted it gets parked in relevant headers of information which can later be matched with relevant job openings based on skills, location, education, or experience.
5. Current Connections: This is one of the lesser technical methods that can be equally efficient in certain scenarios, especially when the target prospect is to be found from a smaller/limited pool. To apply this method effectively it requires hash mapped data of the whole pool of prospects. If an applicant has 2-3 connections already working in the employing organization, he is considered as a viable option. Jobs matching the applicants’ skill set can be recommended using a combination of LSH and KNN techniques. All the leading methods of job matching discussed above can be built using the highly rated services of AWS’s SageMaker Studio, Textract or Comprehend. Azure and Google Cloud Platform’s machine learning and text analytics solutions are also used by industry leaders.

These job recommender systems relate to our project because they all offer ways on how we could have constructed our project. However, for our project, we decided to use TF-IDF values to produce cosine similarity coefficients as well ask nearest neighbors.

**Dataset Description:**

The dataset we used was collected from Kaggle and it contains job postings from Glassdoor.com from 2017-2018 with the following attributes: job title, salary estimate, job description, rating, company name, location, headquarters, size, founded, type of ownership, industry, sector revenue and competitors. There are about 741 different job descriptions in this dataset so it is safe to assume that we have ample amount of documents to compare to our user’s submitted resume in order to find the best job recommendation.

**Design & Methods:**



Above is an illustration of the content based system meant to reciprocate existing job descriptions with one that a user would like to compare against to find the most similar ones.

The design of our recommender system features different ways of classifying the job descriptions in the jobs\_data set with the person’s pre-processed and tokenized resume.

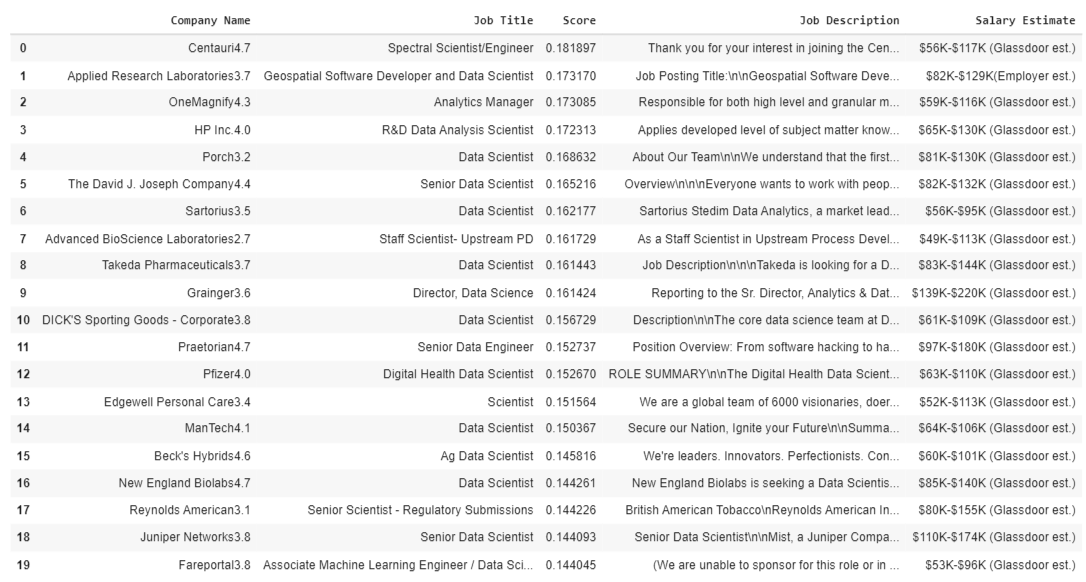
1. The user sends a user query which is their resume, the number of jobs they want
2. Initialization: The pre-processing steps occur where the resume is tokenized, lemmatized, and cleaned before being put into a TF-IDF Vectorizer that counts the number of occurrences of each word in the resume before hand, the same Vectorization is done for the job description column of the jobs dataset
3. Classification: Comparison is done by using the TF-IDF values to produce **cosine similarity** coefficients as well as **k nearest neighbors.** Only values above the 10% threshold are considered
4. Doc Source: Original dataset is referenced and entire job objects are returned from previous steps saved by job index and similarity scores
5. User Feedback: Done by feedback form on frontend of web application, meant to make improvements for system (discussed in the future improvements section)

**Experiments & Results:**

Our example uses two generic resumes, one particularly tailored for Data Science Positions at the senior level (Resume2) and another for general Computer science ones (Resume). This is done to show that the relevance for each of the outputs works as the Data science based resume picks relevant jobs to Data Science Senior level positions, and the Less specific resume returns leading jobs more so based on Computer Science generally or similar job description based ones rather.

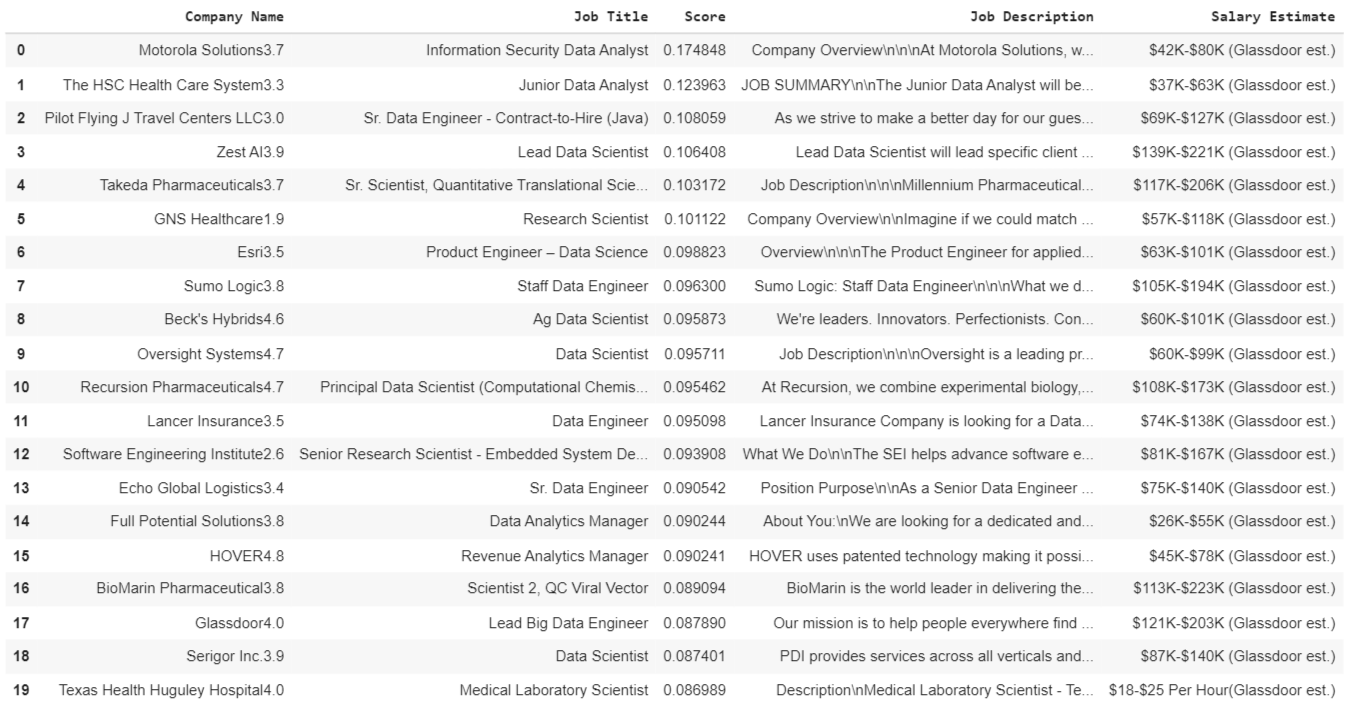
Cosine SImilarity for Experienced Data Scientist Results:





Cosine SImilarity for Computer Scientist Student Results:





The Majority of our experiments consisted of what was learned from our other classes that deal with SKLearn packages including other methods for similarity comparison. Namely from CSCI 355 Artificial Intelligence, we must credit for our understanding of how these SKLearn packages work, particularly the KNN model, as well as what we learned from trial and error during CSCI 626 (this class) homeworks in terms of Cosine Similarity and TF-IDF.

Our point of comparison between the two job predictions is that we can use the fact that the jobs predicted are the same for both KNN and Cosine Similarity. Namely for the Cosine Similarity, we can see that the comparison scores are at 18% max, which is about what we can expect for a resume being compared to the job descriptions, especially, not having tokenized and cleaned the data in the same way for the job descriptions as the resume itself.

**Future Improvements:**

Some future Improvements that we would like to do in the future or if time permitted are:

* Consider and compare more attributes for similarity
  + Ex: Take in the job titles, location, other attributes that exist in dataset
* Find a way to improve the result of the recommendations by weighing values for KNN and cos sim
* Improve the feedback system
* Adding Performance evaluations

**Conclusion:**

After constructing our job recommendation system, we were able to recommend our user with the top 20 jobs out of our dataset using a user’s query through TFIDF, while utilizing the SKLearn algorithms of cosine similarity and K Nearest Neighbors in order to find similarity scores. To build on this, we constructed a UI for our project using Anvil. However, while trying to link the Anvil UI with our Jupyter Notebook, we were unable to connect the data with the Anvil UI since the Anvil UI was unable to display our data that were stored in the Pandas framework. Thanks to this course project, we were able to learn how to apply TF-IDF, cosine similarity, and K Nearest Neighbors in order to construct a information retrieval system as well as build on our current knowledge of Python.

**Team Contributions:**

Jeffrey Aguilar & Harrown Kenta:

* Developing the Job Recommendation System on the Colab Notebook.

Jeffrey Aguilar & Tahura Islam:

* Constructing the Anvil UI & working on connecting the UI and data input from Anvil with the Colab Notebook.

**References:**

1. References to learning model training and outputting top jobs: <https://towardsdatascience.com/a-marijuana-recommendation-system-using-tf-idf-and-k-nn-bd472242cb3f>
2. <https://de-drishti.medium.com/personalized-job-recommendation-system-using-collaborative-filtering-6d663046692a>
3. Similar Project Idea handling dataframes:<https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/>
4. CSCI 355 Artificial Intelligence & CSCI 626 Information Retrieval general course material
5. Similar Job Recommendation Systems: <https://www.makeen.io/5-job-recommendation-systems-via-machine-learning-techniques/>
6. DataFrames: <https://www.geeksforgeeks.org/python-pandas-dataframe-at/>