# AIE425 Intelligent Recommender Systems, Fall Semester 24/25
# Course Project: Job Recommendation System

**Karim Mamdouh, ID: 222102535**
**Maram Elbana, ID: 222102387**

# Table of Contents

# Introduction

This report presents the design and implementation of a Job Recommendation System, a content-based system that recommends jobs based on job titles, roles, and descriptions. Based on textual similarities between job descriptions and titles, the system effectively matches job seekers with relevant job openings.

The recommender system analyzes job data, including titles, positions, company names, and descriptions. Cosine similarity is used to measure the textual similarity between job descriptions, enabling job recommendations based on the highest similarity scores. This approach is classified as content-based filtering because it relies solely on the textual content of job postings, rather than user interaction data.

Preprocessing steps, such as tokenization, stemming, TF-IDF vectorization, and missing value handling, prepare the dataset for the recommendation engine. The system then calculates similarity scores to recommend jobs based on user-provided inputs.

# Data Collection and Preprocessing

The dataset used for this project is sourced from Kaggle and consists of 84,090 rows and 23 columns. However, only the following key attributes were selected for the recommendation engine:

- Status: Indicates the current job status (e.g., open, closed).
- Title: Represents the job title (e.g., Software Engineer, Data Analyst).
- Position: Specifies the job position (e.g., Senior, Junior).
- Company: Represents the company offering the job.
- Job.Description: Describes the requirements and responsibilities of the job.

For a more manageable analysis, a random sample of 1,000 entries was selected from the dataset.

## Preprocessing Steps

1.Handling Missing Values: Missing values in the dataset were filled with the most frequent value (mode) in each column to maintain consistency and avoid data loss.

2. Data Cleaning: Non-alphanumeric characters were removed, and text was tokenized into individual words. Stemming was applied to reduce words to their root forms (e.g., "running" becomes "run"), and stopwords were removed to retain only meaningful text.

3. Text Vectorization: Using the TF-IDF technique, the cleaned text was converted into numerical vectors. This representation highlights the relative importance of each word in the dataset.

4. Cosine Similarity: Cosine similarity was calculated between job descriptions and titles to determine their textual similarity. The higher the similarity score, the more relevant the recommendation.

# Background on Cosine Similarity

Cosine similarity is a widely-used metric for measuring the similarity between two non-zero vectors in a multi-dimensional space. It is particularly effective for comparing text data in applications such as information retrieval, document clustering, and recommendation systems.

The formula for cosine similarity between two vectors A and B is:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|}$$

Where: - $A \cdot B$ is the dot product of the two vectors. - $\|A\|$ and $\|B\|$ are the magnitudes (norms) of the vectors A and B.

In simple terms, cosine similarity computes how much two vectors point in the same direction, regardless of their magnitude. The result of this calculation is a value between -1 and 1: - 1 means the vectors are identical (pointing in the same direction). - 0 means the vectors are orthogonal, or have no similarity. - -1 means the vectors are diametrically opposed (pointing in opposite directions).

- Application in Text Analysis

Cosine similarity is particularly useful in the context of text analysis, where it can be applied to determine the similarity between documents or text-based entries (such as job descriptions or job titles).

In text mining, documents are typically represented as vectors in a multi-dimensional space, where each dimension corresponds to a term or word in the document. The most common method for creating these vectors is TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. TF-IDF assigns a weight to each word based on how frequently it appears in a particular document and how rare it is across all documents in the dataset. This helps in emphasizing important words that are more relevant in a given document.

Once the documents are converted into TF-IDF vectors, cosine similarity can be used to compare them and determine how similar two job titles or job descriptions are based on their textual content.

- Advantages and Disadvantages of Cosine Similarity

Advantages: - Efficient and scalable for large datasets. - Robust to length differences in vectors. - Easy to interpret; higher scores indicate more similarity.

Disadvantages: - Does not capture the deeper semantic meaning of words. - Only considers textual content, which may not always be sufficient for more accurate recommendations.

# The Design of the Recommender Engine in Job Recommendation Domain

The job recommendation engine operates as a content-based filtering system, focusing on the textual attributes of job postings to generate recommendations. Key design components include:

1. **Data Selection**: Attributes such as 'Status', 'Title', 'Position', 'Company', and 'Job.Description' were selected for the recommendation system.

2. **Preprocessing**: Missing values were handled, and text was cleaned by removing irrelevant characters, tokenizing, stemming, and removing stopwords.

3. **Feature Engineering**: A new column, `clean_text`, was created by combining 'Job.Description', 'Title', and 'Position'. This column was used for similarity calculations.

4. **Text Vectorization**: The TF-IDF technique transformed the cleaned text into numerical vectors for analysis.

5. **Similarity Calculation**: Cosine similarity was used to compute the similarity between job descriptions and user input.

6. **Job Recommendation**: Recommendations were generated based on the most similar job titles, as determined by cosine similarity scores.

7. **User Interaction with Streamlit**: Streamlit was used to create an interactive user interface, allowing users to input job titles and receive recommendations in real time.

# Description of the Recommender Engine Implementation, Tools, and Libraries

The implementation of the recommendation system was done in Python using the following libraries:

- **Pandas**: For data manipulation and handling missing values.
- **NLTK**: For text cleaning, including tokenization, stemming, and stopword removal.
- **Scikit-learn**: For TF-IDF vectorization and cosine similarity calculation.
- **Pickle**: For serializing the processed data and similarity matrix.
- **Streamlit**: For building an interactive user interface for job recommendations.

The workflow involved:

1. Loading and preprocessing the dataset using Pandas.
2. Cleaning the text using NLTK and regular expressions.
3. Converting text to TF-IDF vectors with Scikit-learn.
4. Calculating similarity scores using cosine similarity.
5. Serializing the data and model with Pickle for future use.
6. Creating a user-friendly interface with Streamlit for real-time recommendations.

# Results and Visualization

The results of the job recommendation system include various insights and visualizations that demonstrate the system's ability to analyze and recommend relevant jobs based on textual similarity.

One key visualization generated during the analysis phase is a 3D scatter plot of the job data, which highlights the relationship between job positions, companies, and job statuses. The plot allows us to better understand the distribution and clustering of job postings based on their attributes.

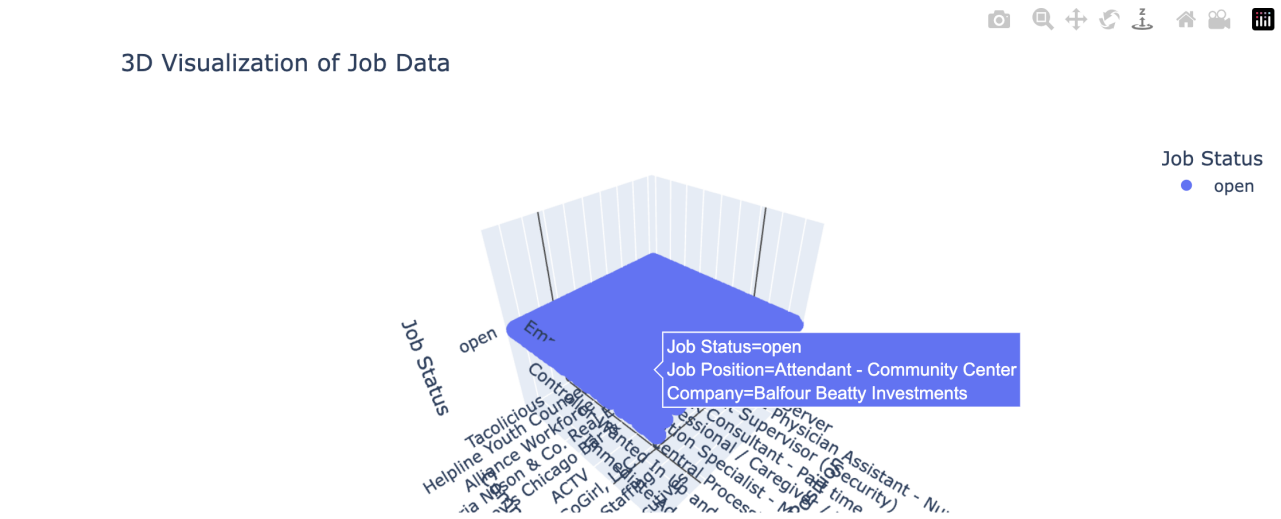


Figure 1: 3D Visualization of Job Data. This scatter plot shows the relationships among job positions, companies, and job statuses. For example, the highlighted point corresponds to a job posting for a "Stockroom Assistant - Part-time" position at "Staff Right Services LLC" with the status "open."

The visualization was created using the Plotly library, which enabled interactive 3D rendering of the data. Key observations include:

- Clusters of job positions across specific companies and statuses.
- Visual identification of open positions in relation to companies.
- Enhanced understanding of the data distribution for analysis.

This 3D visualization provides an intuitive way to explore the dataset and helps users identify potential patterns in the job market, which can inform the recommendation logic.

The job recommendation system produced two sets of results based on user inputs, showcasing the effectiveness of the model in identifying relevant job postings using content-based filtering. Below are the outputs with comments:

## Example 1: Searching for "HHA CNA Need Immedi Bayada Home Health Care"

The system returned a list of jobs that match the query, focusing on similar roles within the same organization and related descriptions.

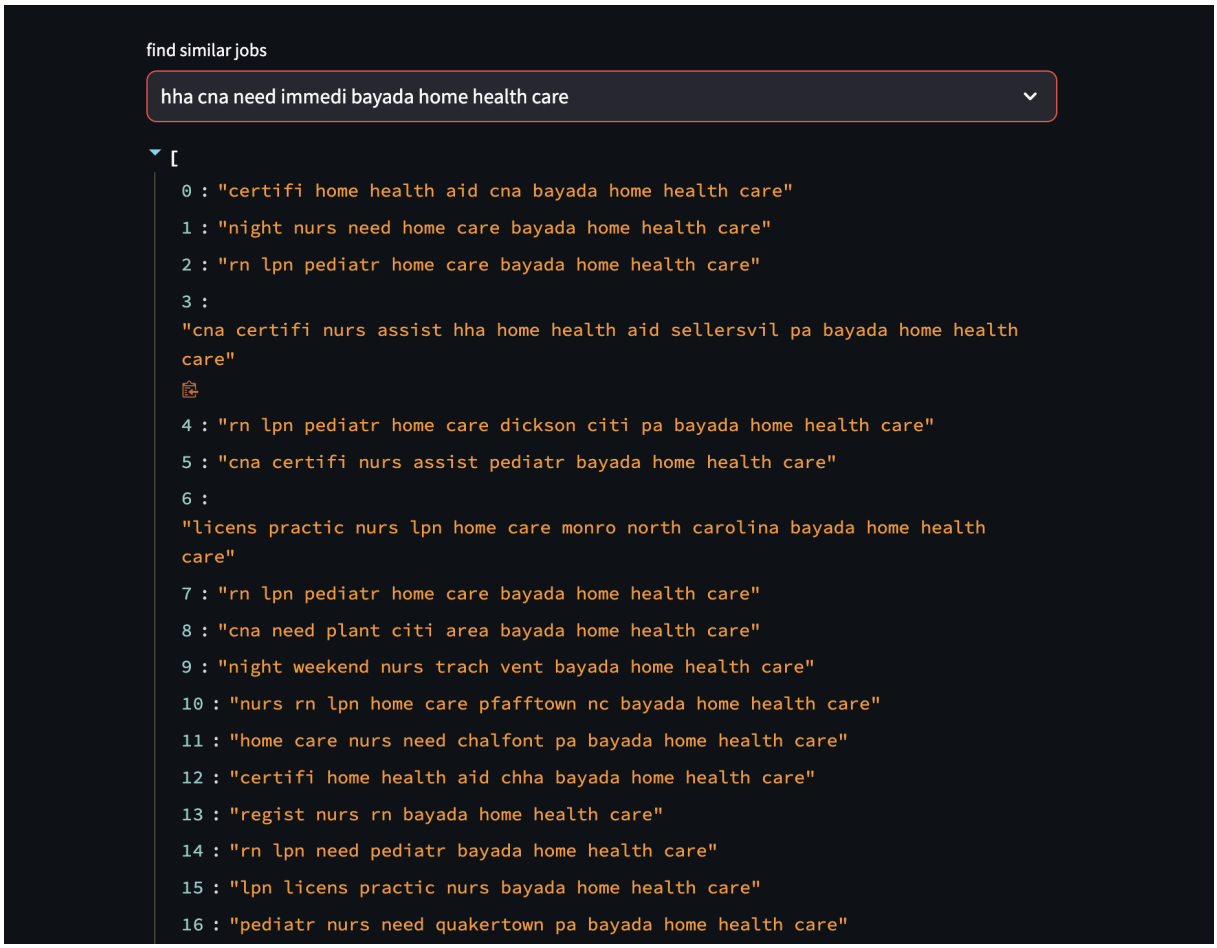


Figure 2: Job Recommendations for Query: "HHA CNA Need Immedi Bayada Home Health Care". The recommendations effectively highlight positions like "Home Health Aid" and "Night Nurses" that match the query's context.

## Example 2: Searching for "Office Assist Officeteam"

The system returned job recommendations relevant to administrative and clerical roles, emphasizing jobs with "Office Assist" or related titles.

## Comments on Results

1. The model successfully identifies jobs with high textual similarity to the user queries. 2. The recommendations are specific and contextually relevant, as shown in the examples, where the outputs align with the job descriptions and organizations queried. 3. Some repeated entries indicate a potential improvement area to refine the output for diversity in recommendations. 4. Overall, the system demonstrates strong performance in matching job titles and descriptions using cosine similarity and TF-IDF vectorization.

These examples validate the effectiveness of the content-based filtering approach and illustrate the practical usability of the model in real-world job recommendation scenarios.

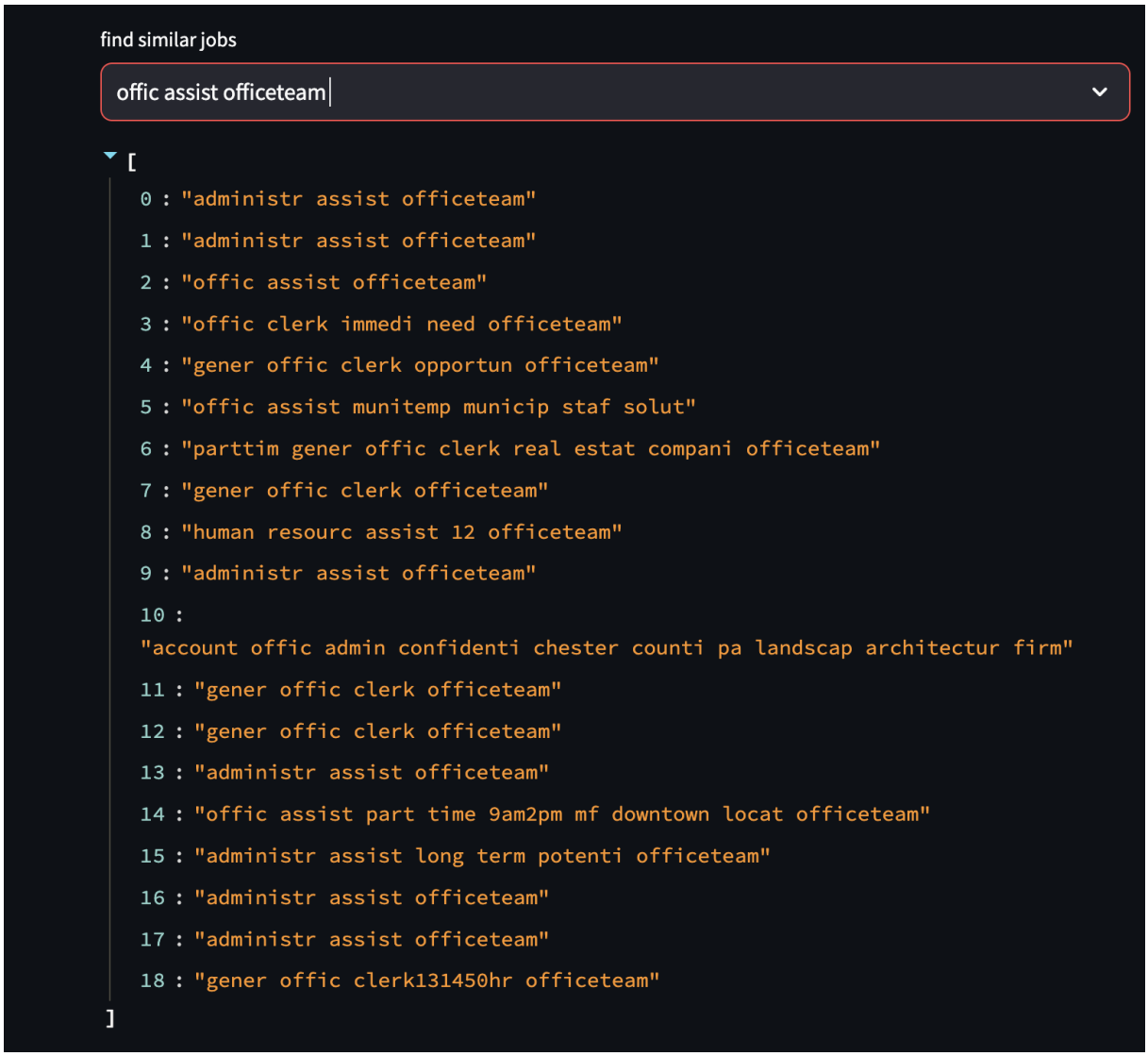


Figure 3: Job Recommendations for Query: "Office Assist Officeteam". The recommendations show closely related roles like "Administrative Assistants" and "Office Clerks" across similar job contexts.

# Conclusion

The job recommendation system developed in this project demonstrates the effectiveness of content-based filtering techniques in matching job seekers with relevant job opportunities. By leveraging textual data such as job descriptions, titles, and positions, the system provides personalized recommendations based on the similarity of job postings to user queries. The use of TF-IDF vectorization and cosine similarity ensures that the recommendations are contextually accurate and relevant.

The visualization tools used in the project, such as 3D scatter plots and interactive interfaces via Streamlit, further enhance the usability of the system, making it accessible and easy to interpret for end-users. Overall, the system is a practical and efficient solution for job recommendation tasks, with potential for real-world applications in recruitment platforms and job search engines.

# Enhancements and Future Work

While the system achieves its primary objective, there are several areas for potential improvement and expansion: 1. Incorporating User Feedback: - Currently, the model operates solely on content-based filtering. Integrating user feedback, such as ratings or interactions, could enable hybrid recommendation approaches that combine content-based and collaborative filtering techniques.

2. Improving Diversity in Recommendations: - The current system sometimes returns repetitive or highly similar results. Adding mechanisms to ensure diversity in recommendations can enhance user satisfaction.

3. Handling Synonyms and Semantic Similarity: - Incorporating advanced natural language processing techniques, such as word embeddings (e.g., Word2Vec or BERT), could improve the system's ability to understand synonyms and semantic relationships between words, leading to better recommendations.

4. Scalability: - Optimizing the system for larger datasets and integrating parallel processing techniques can ensure scalability for real-world applications involving millions of job postings.

5. Real-Time Updates: - Implementing mechanisms to dynamically update the dataset with new job postings and automatically retrain the model can keep the system relevant and up-to-date.

6. Personalization: - Including user profiles and preferences can enable more personalized recommendations, tailored to individual career goals and aspirations.

7. Enhanced Visualizations: - Further improvements in data visualization, such as clustering job roles or mapping job categories, could provide more insightful analyses for recruiters and job seekers.

By addressing these areas, the system can evolve into a robust and comprehensive job recommendation engine capable of meeting diverse user needs and adapting to changing job market trends.

# References


1. Streamlit Documentation, [Online]. Available: https://docs.streamlit.io/.
2. "Cosine Similarity," ScienceDirect, [Online]. Available: https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=Cosine%20similarity%20measures%20the%20similarity,document%20similarity%20in%20text%20analysis..
3. K. Kandij, "Job Recommendation Datasets," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/kandij/job-recommendation-datasets.
4. P. Gaurav, "Step-by-step content-based recommendation system," Medium, [Online]. Available: https://medium.com/@prateekgaurav/step-by-step-content-based-recomm~:text=1,their%20viewing%20and%20purchasing%20history..
5. "Vectorization in Machine Learning," Comet Blog, [Online]. Available: https://www.comet.com/site/blog/vectorization-in-machine-learning/#:~:text=Vectorization%20is%20the%20process%20of,training%20time%20of%20your%20code..
6. "Pickle Python Tutorial," DataCamp, [Online]. Available: https://www.datacamp.com/tutorial/pickle-python-tutorial.