National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name: Srikanth Vunnam

Student ID: X23297271

Programme: MSc in Artificial Intelligence

Year: 2024 - 25

Module: MSc Research Practicum

Supervisor: Devanshu Anadh

Submission Due Date: 11 - Aug - 2025

Project Title: Transforming Job Search with AI- Recommendation Algorithim

Word Count: 8247

Page Count: 25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Srikanth Vunnam
Date: 10- Aug- 2025

Attach a completed copy of this sheet to each project (including multiple copies)	**Yes**
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	**Yes**
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	**Yes**

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

CAREER REACTANT: TRANSFORMING JOB SEARCH WITH AI-ENHANCED RECOMMENDATION ALGORITHMS

Forename Surname
Student ID

Abstract

This paper introduces a detailed description of Career Reactant, an AI-controlled hybrid job recommendation platform that promotes greater individuality, transparency, and ethical conformance in online job matching. Conventional systems can perform based on an anti-keyword tree and do not make use of user context, hence giving irrelevant or biased suggestions. To overcome these drawbacks, the paper will combine content-based filtering, collaborative filtering, and deep learning with PyTorch, and NLP will be applied and used to suggest skills. Experiments were conducted over publicly-available datasets and model performance was captured by technical measures like the F1-score and the AUC-ROC and also through user-centred measures like the click-through rate. Ethics was also incorporated into the product in terms of data privacy, data bias, fairness-aware design and adherence to GDPR. The learning system was implemented on an interactive Gradio interface where real-time feedback and immediate adjustment of the model was possible. The results substantiate the conclusion that Career Reactant enhances the relevance and transparency of recommendation, providing a scalable (and user-specific) method of addressing the new terrain of digital employment.

1 Chapter 1: Introduction

1.1 Problem Statement

Static keyword matching is a common method of traditional job recommendation systems, based on several key findings: the employment market is dynamic and matching is influenced heavily by the interaction between the employer and potential candidate, reliance solely on keywords neglects the finer details and the changing abilities of job applicants. Such systems are impersonal, inflexible, and not transparent, often making irrelevant or biased job recommendations. Moreover, ethical issues like algorithmic bias, privacy, and recruitment fairness lack enough consideration throughout them. With the job market becoming more competitive and data-driven, the critical requirement is the availability of intelligent systems that can give precise, user-specific, and ethics-compliant recommendations (Alsaif *et al.* 2022). This study will overcome these disadvantages by creating a hybrid approach to AI-enhanced employment-related tasks: a Career Reactant job recommendation system that combines deep learning, NLP, and immediate user maybes-based response to provide more relevant and equal job suggestions.

1.2 Aim

This research aims to develop and evaluate a hybrid AI-based job recommendation system. Career Reactant is a personalised, fair, and dynamic job recommendation framework centred on content-based recommender systems, collaborative filtering, and deep learning algorithms, with consideration of ethical issues in algorithmic recruiting.

1.3 Objectives

The following objectives have been analysed in this research:

- To analyse the state of the art in terms of AI-based job recommendation systems regarding personalisation, fairness, and user experience.
- To conceptualise and develop an AI-based recommendation model that is a hybrid between TF-IDF and collaborative filtering, and uses Deep learning with PyTorch.

- To use natural language processing (NLP) to capture offer skill and job semantics, and hence improve job user matching.
- To measure the effectiveness of the system according to the technical (F1-score, AUC-ROC) and individual (click-through rate) criteria.
- To enhance ethical and legal issues concerning algorithmic bias, data privacy of user data, and data transparency with fairness-aware models and the enforcement of the GDPR.

1.4 Research Question

"What steps can a key combination of AI and human-based recommendation engine take to empower the job search platforms to be more individualised, non-discriminative, and effective and save it from ethical and legal challenges?"

1.5 Research Rationale

Job market digitalisation, which is taking place more speedily, has expanded dependency on websites in search of job providers. Nonetheless, current job recommendation systems are in many cases inadequate in providing precise, personalised, and non-biased outcomes. They usually rely on the keyword-based filtering, which ignores user preferences, the evolution of skills and contextual importance (Javed *et al.* 2021). Also, issues of discrimination, lack of transparency and privacy of user data have not been addressed in most systems. With the further development of artificial intelligence, natural language processing, and deep learning, it is possible to rethink the process of job recommendations generation. Such research is topical because it helps overcome existing problems of the current systems by creating Career Reactant, a hybrid AI-supported model based on content-based and collaborative filtering, using fairness-aware algorithms. It can benefit the field due to a scalable, ethical, and user-tunable system that is more likely to fit the contemporary labour requirements and endorse equal access to six employment prospects.

1.6 Methodology Outline

In this research, a Design Science Research (DSR) approach is used to develop and test the Career Reactant system. The given hybrid recommendation engine is developed on the basis of content-based filtering, collaborative filtering, and a deep learning solution applied via PyTorch. NLP-based techniques are used to extract features related to jobs and people based on SpaCy. Kaggle offers public datasets that have been preprocessed and modelled using Python on Google Colab. The results of TF-IDF, LightFM, and a PyTorch hybrid are retrieved by means of the interactivity on Gradio. Performance is evaluated with technical measures (e.g., F1-score, AUC-ROC), and with the behaviour of users (e.g., click-through rate). Anonymisation, GDPR compliance, and AI-based detection of bias with the help of AI Fairness 360 belong to the ethical practices. This is a cyclic procedure enabling live learning and dynamic employment fitting.

2 Chapter 2: Literature Review

2.1 Chapter Introduction

In this chapter, the current literature on AI-supported job recommendation systems is closely examined, and the differences between technology and ethics are highlighted. Since more people compete for jobs today, generic search methods do not offer the targeted or complementary opportunities that are needed. They focus on recent improvements in artificial intelligence such as machine learning, natural language processing, and generative AI, meant to improve the accuracy and fairness of job search platforms. Example papers came from various fields, such as AI in recruitment, personal learning analysis, and cutting down bias in hiring, which helped guide the development of the Career Reactant framework. They highlight both the positive aspects and challenges of present-day AI applications by pointing out that real feedback, ethical approach, and user design are important. It helps to point out gaps that are important for shaping the research, leading to a better and fairer job recommendation service.

2.2 Empirical Study

2.2.1 AI-Enhanced Learning Analytics for Career Decisions

The research by **Gedrimiene *et al.* (2024)** examines how utilizing AI in learning analytics can assist students during the process of making career-related decisions. The research examines how young people interact with digital tools, mainly focused on moments of career decision-making. Based on reports of 106 participants who worked with an LA tool, the researchers looked at how they accepted the technology through TAM and how they made decisions using CDM. It was found that the AI tool helps by giving career-related guidance, directions, reflective assistance, and new perspectives on possible career paths. On the other hand, the research reveals that there isn't enough context in the assessment, it's not well personalized, and students receive support disproportionally during different stages of their career journey. If the system did not have real-time updates and offered more flexible tailoring, it would be more effective for people who want personalized tips for their careers. It is very important for the development of Career Reactant because it reinforces that incorporating personalization, repeated feedback, and users' information is key for effective job recommendation systems. Even though the study is largely based on what people say about themselves, it gives valuable information about how current AI tools function in the field of career guidance.

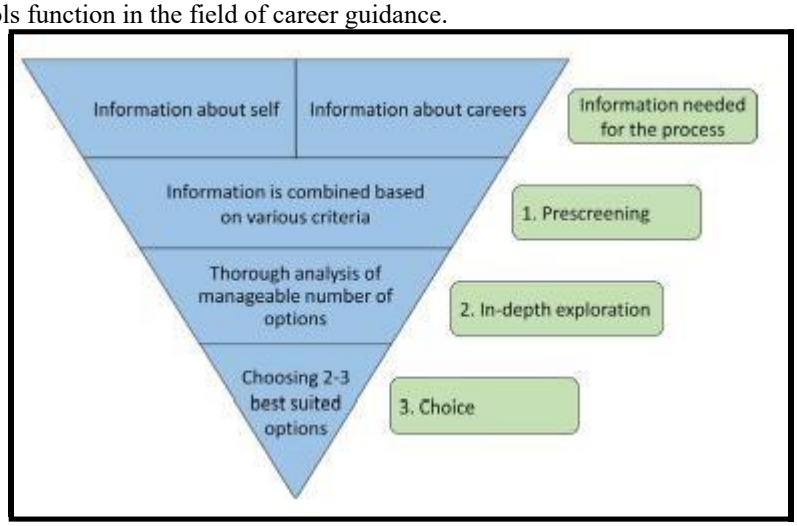


Figure 1: The CDM and the information needed to make the decision
(Source: Gedrimiene *et al.* 2024)

2.2.2 Media 2.0: AI-Enhanced Communication and Content Creation

The work of **Singh (2024)** carefully discusses how artificial intelligence is shaping the future of media, communication, and content creation known as Media 2.0. AI is described in the text as a key feature in automated news, musical creation, sorting recommendations, and tailoring content. It also points out that marketers now depend more on AI to better understand and target their audience. A major topic is the ethics of AI, mainly due to issues such as privacy, transparency in algorithms, and bias. Another point made is that future media will likely include storytelling combined with virtual and augmented reality technologies that use AI. Though the paper is rich in ideas, most of its content is theoretical and does not look closely at particular AI systems or algorithms. It focuses on areas other than recruitment and job search, but the topics of personalization, ethics, and user involvement give useful insights for improving Career Reactant. Such insights can help integrate AI into job recommendation sites in a fair manner and keep users confident in the service.

2.2.3 AI-Enhanced Recruitment and Diversity in Finland

In her study, **Ashik (2023)** looks into how AI-powered recruitment tools influence diversity and inclusion in Finland. The study looked at data from 109 participants from various groups to evaluate how comfortable they are with AI, the biases they perceive in AI, and the inclusive quality of recruitment practices. The research shows that, while AI is meant to help avoid human bias in hiring, people shared concerns about fairness, mainly from Finnish non-native applicants facing challenges due to limited language skills and a different background. It points out that, despite sharing many recruitment activities to improve efficiency with AI, there is a risk that these tools could strengthen the same kinds of unequal practices. A notable point is the way the study looks at user experiences in actual labor market situations, letting us understand the effects of AI in recruitment. Yet, because the study only covers Finland and relies on what people say, the results may not be very reliable. Notwithstanding these shortcomings, it confirms that systems and laws must be fair to avoid ethical issues in recruitment. This finds significance for Career Reactant, specifically in ensuring algorithmic fairness, is aware of cultural influences, and frequently checks for inclusivity within job recommendation tools.

2.2.4 Enhancing Education with NLP through AI-Enhanced Q&A Evaluation

In their paper, **Ahamed *et al.* (2024)** describe an AI learning platform meant to raise educational standards using advanced techniques for producing questions and answers, judging answers, and offering personalized recommendations. The authors use different features, such as identifying weak areas, repeating tests, support forums, and expert assistance, to ensure students learn well. Significantly, it handles functions such as auto-generating questions and assessing responses of students in real time, from the content provided, to create custom and speedy suggestions. The thorough approach encourages students to engage, remember things, and check their progress. Even though the focus is on education, personalized feedback, using NLP to read texts, and continuous performance measures can be used in job recommendation systems. For instance, similar methods used to find student weaknesses might be useful to find skills that a job seeker lacks and match them to suitable opportunities. Yet, because no empirical measures, results from user testing, or comparisons with current educational tools are included, its practical use is not as convincing. In addition, matters such as keeping learner data secret remain unresolved, which could cause issues in broader situations. Although the main goal of the study is to enhance learning, the methods it uses serve as a strong base for systems like Career Reactant. The article brings new insights to AI personalization and highlights the value of user feedback in boosting system recommendations. Such ideas can usefully influence how AI supports job search.

2.2.5 A Comprehensive Review of AI Techniques for Bias in Job Hiring

Albaroudi *et al.* (2024) thoroughly explore available methods that help overcome algorithmic bias in selecting people for jobs. Since AI plays a big role in CV review and automating recruitment, the study indicates concerns about its fairness and how it can lead to discrimination. The authors study and review various studies and examples to find ways to remove bias, including vector space correction, adding more data, and using models that prioritize fairness. A major effort is put into natural language processing (NLP) and deep learning, as they are promising tools for finding and addressing bias in recruitment data. According to the study, being open about what algorithms do, involving humans and AI, and other ethical principles are important for fair hiring algorithms. The report includes plenty of useful discussion and ideas, but it does not do much testing or comparisons that would be typical in empirical studies. Furthermore, it is not satisfactory in resolving practical issues in deployment, including company resistance, newly emerging biases, and sticking to global standards of ethics. Nonetheless, the paper's results benefit Career Reactant as it strives to provide job suggestions that are inclusive to everyone. The use of fairness-aware AI and bias-mitigation approaches allows Career Reactant to design a system that recommends jobs fairly and helps level out job-access opportunities. This paper illustrates that both progress in technology and ethical actions need to occur together in, for instance, systems that decide on employment.

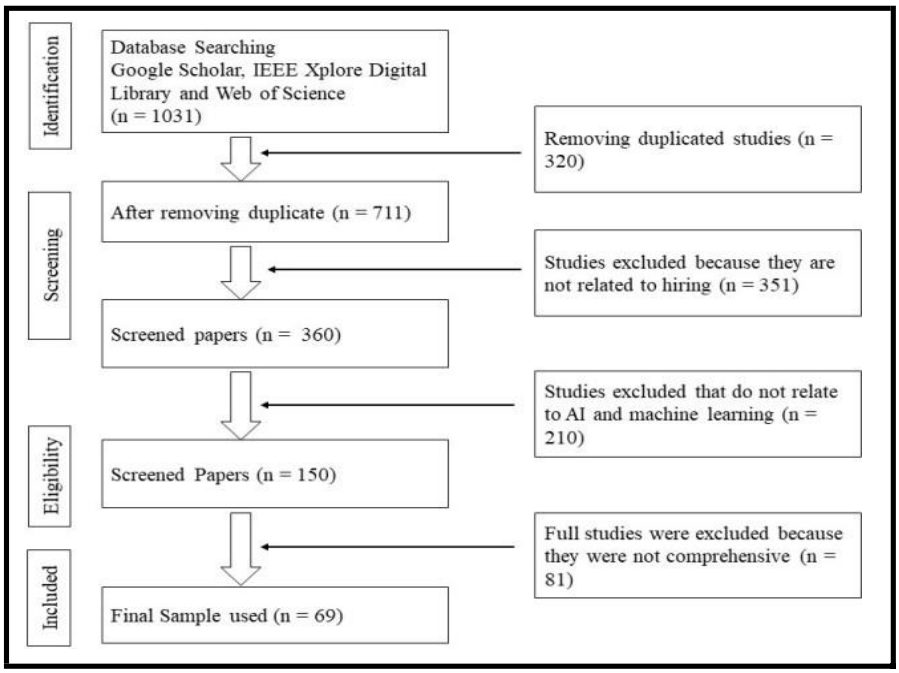


Figure 2: Description of the Literature Screening Process by Albaroudi *et al.* (2024)
(Source: Albaroudi *et al.* 2024)

2.2.6 Applying Generative AI in the Job Finding Process

Sivasubramaniam (2024) examines how generative artificial intelligence can assist job seekers by ensuring a user-friendly job portal offers them relevant and reliable job suggestions. According to the thesis, present-day job portals regularly provide inappropriate matches because they depend on simple listings that do not focus on what individuals are truly looking for. According to the study, a generative AI framework could help job seekers by having them input their aims and capabilities, after which the system generates tailored job proposals based on their input. The research stands out by supporting job seekers in choosing and customizing what they receive, all with the help of AI and easy-to-understand language. Still, the research is only theoretical and does not have a practical prototype or plan for implementation. Since the authors did not employ empirical tests or assess systems, it is difficult to determine if the model can be used in practice. Ethical design, protecting personal data, and dealing with biases are minor concerns highlighted in the book. Even with these problems, the thesis brings attention to how Career Reactant benefits from being creative, allowing users to control, and being flexible for diverse scenarios. It encourages AI to match users to positions not only through existing jobs, but also by considering both their preferences and what is happening in the labor market. This work's concept fits well with Career Reactant's goal to make job search more dynamic and depend on AI rather than staying the same for users.

2.3 Literature Gap

While work is being done on using AI in education, media, and hiring, there is still a big need to develop online job recommendation tools that are personal, fair, and adjustable at any time. The usual approach is to match keywords without adapting to what users type in inches after time. Besides, people consistently recognize ethical matters, yet few solutions are put in place to resolve them. Generative AI and job search optimization have not been extensively researched. By designing an AI framework that is user-friendly, hybrid, and aware of ethics, the Career Reactant project intends to cover these gaps.

2.4 Theories and Models

The system is created following important theories that support recommending intelligently, keeping users engaged, and using AI ethically. The CBF approach is a basic technique where content is filtered to support item recommendations based on the user's preferences. It looks at both job descriptions and resumes to locate matching phrases. In addition, Collaborative Filtering (CF) uses the interactions of those with similar preferences to come up with personal suggestions (Koren *et al.* 2021). Still, these approaches come with weaknesses: CBF does not handle a variety of tasks properly, and CF deals with the cold-start obstacle. That's why more and more researchers are turning to hybrid deep learning methods.

Using neural networks, NCF is designed to improve the prediction of user-job relationships when compared to traditional methods (Li and Kim, 2021). Also, using Natural Language Processing models like SpaCy helps to study the content of resumes and job posts more closely for better matching.
The Career Decision-Making (CDM) model is especially useful for decision-making in this area (Wang *et al*. 2022). The system aims to help people during self-assessment, exploration, and decision-making using adaptive feedback loops.
Like IBM's AI Fairness 360, ethical use of AI involves making things clear, responsible, and less likely to cause prejudice. All these theories and models influence the design of Career Reactant by providing guidelines for recommendations, personalization, fairness, and learning what users interact with. With these principles combined, the system intends to offer effective and fair job suggestions.

2.5 Conceptual Framework

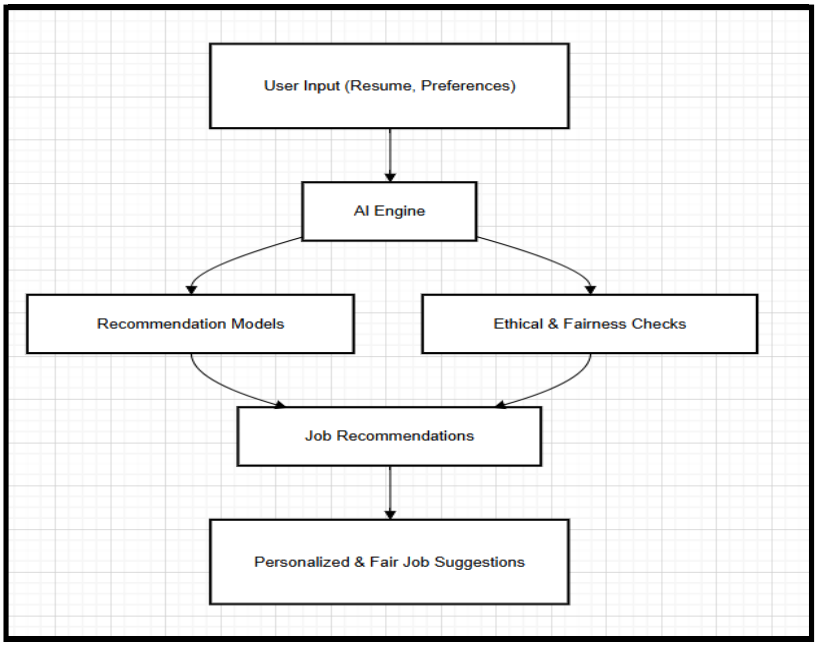


Figure 3: Conceptual Framework for model development
(Source: Self-made)

2.6 Chapter Summary

Main insights about AI in job recommendation, recruitment, education, and media were explored, covering successes and ongoing difficulties. Although AI may help with personalization and efficiency, the systems now in place are not always fast enough, fair, or user-friendly. By reviewing the studies, the researchers can note that using content-based and collaborative filtering techniques, as well as deep learning and NLP, can enhance the matching of candidates to jobs. Issues such as bias in algorithms and protecting people's data get little attention. Models such as CDM and tools such as AI Fairness 360 greatly improve our decision-making in AI. The ideas discovered help guide the development of the Career Reactant system.

3 Chapter 3: Methodology

3.1 Chapter Overview

This chapter describes the design, development, and assessment experiment procedure implemented to produce the AI-filled job suggestion tool, Career Reactant. It provides a structured approach to the design of systems, the execution of models, and the processing of data. It entails an integrated approach to recommendation strategy, or a hybrid algorithm that will bring together content-based and collaborative filtering, with support in deep learning and natural language processing. This project utilizes publicly available datasets on Kaggle as the source of data and manipulates and analyzes them in the Google Colab environment, in Python. This chapter also describes the methodology, criteria of evaluation, and the ethical principles according to which the project is conducted and makes it responsible and efficient to shape.

3.2 Research Design and Approach

The methodology of the Design Science Research (DSR) approach is used in the implementation of this project as it is aimed at creating a new artefact to realise the solutions to the identified pragmatic problems (De Sordi, 2021). The relevant application of DSR with the consideration of the related goal of developing a personalized job recommender based on AI is due to the possibility in DSR to systematically create, repeatedly test, and practically validate the Career Reactant system. The proposed new recommendation model will use Content-Based Filtering (CBF) and Collaborative Filtering (CF) which is to be supplemented with Natural Language Processing (NLP) and intensive learning (Widayanti, 2023).. The CBF is based on the similarity between the semantic profile of users and job descriptions, whereas the CF is based on capturing patterns of user interactions to reveal latent preferences (Han *et al.* 2024). The goal of these two methods together is to increase the relevancy and individuality of job recommendations. The system development cycle encompasses data collection, preprocessing, model training, real-time feedback recombination and iterative improvement. It is planned to use Kaggle to divide publicly available datasets and process them in Python with Google Colab. The system will allow continuous user feedback to retrain the model to build an adaptive recommendation engine. This study, however, has a quantitative assessment approach, technical measurements, like the F1-score and AUC-ROC, as well as user-related measures, like the click-through rates. This double concern is the combination of paying both algorithmic accuracy and user satisfaction. As a whole, this strategy would help create a scalable and context-sensitive solution that can be adjusted to the behaviour of users and dynamics of the market, which would mean addressing the ineffectiveness of conventional job recommendation systems.

3.3 System Architecture and Design

The Career Reactant system architecture enables end-to-end personalized job recommendations with the use of AI. It has five significant parts, namely the user interface, NLP processing unit, hybrid recommendation engine, real-time feedback loop, and data storage layer. The modules communicate with each other to achieve a natural flow that would be functional, precise, and flexible. The system begins with the user interface, which has been constructed with the help of Streamlit, where the users can upload resumes, rate job suggestions, and see recommendations. When the resume or profile is received, the NLP module that uses SpaCy will perform the extraction of skills, experience, and related keywords (Jugran *et al.* 2021). Such semantically linked information is compared against job descriptions in similarity vectors similar to TF-IDF and word embedding contexts. The recommendation engine is then created in PyTorch and uses collaborative and content-based filtering (Omar *et al*. 2024). The content-based part assigns the user attributes to job advertisements whereas the more collaborative part sets about identifying user-job interaction patterns as would have been in the past. The hybrid model uses a combination of the two to generate a list of suggested jobs in order of rank. The real-time feedback loop records user clicks, job saves and ratings. This feedback is then used to re-train the model on a dynamic basis so that the system adjusts to the preferences of the players and changes in the labour market. Each of the datasets, such as anonymized user resumes and a list of vacancies, will be stored in a secure SaaS data storage where scalable access to data and model deployment is possible. The modular architecture also creates flexibility such that there is the power to improve the individual components without necessarily disturbing the whole system (Paparistodimou *et al.* 2020).

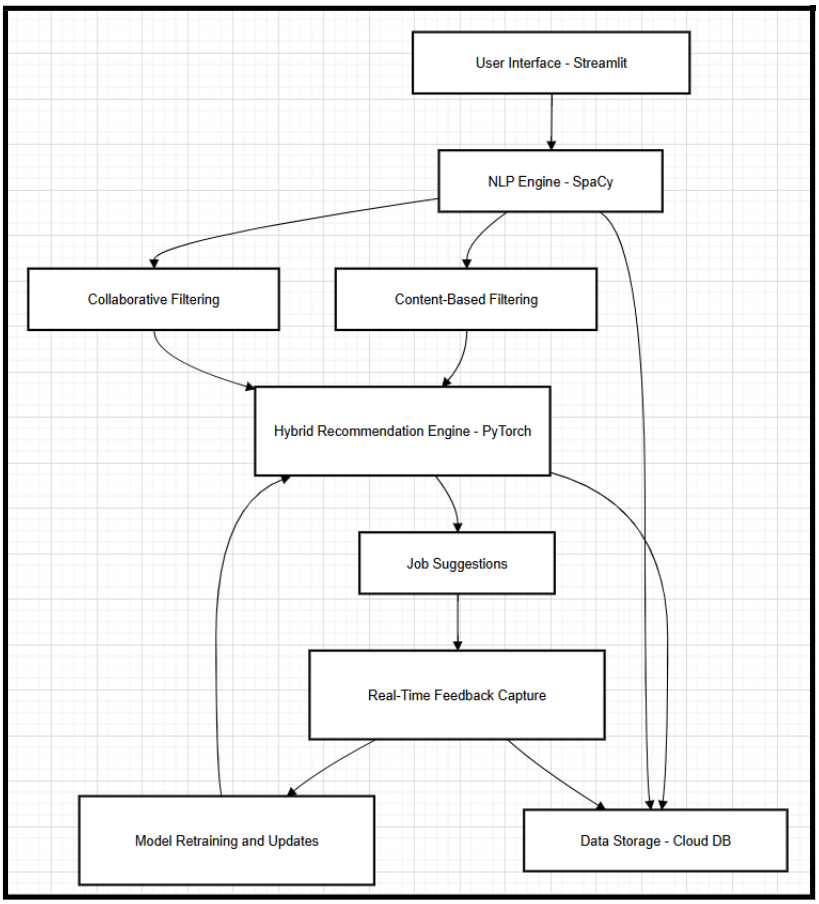


Figure 4: Career Reactant System Architecture
(Source: Self-made)

3.4 Data Collection and Preprocessing

Dataset link- https://www.kaggle.com/datasets/asaniczka/linkedin-data-engineer-job-postings
The publicly available job datasets that can be used as the data of the Career Reactant system are user profiles, job listings, and user-job interactions. Python tools are used to clean and standardize such datasets. Using NLP techniques, text fields of user skills and job descriptions have been converted using tokenization, lowercase, removing stop words, and lemmatizing them to have the same pattern for the semantic matching activities.

In the TF-IDF Content-Based model, the cleaned job descriptions and skills entered by a user are both turned into numerical vectors using a term-frequency inverse document frequency vectorizer. The comparisons between these are through cosine similarity to give recommendations on relevant jobs based on pairs of skill matches.
In the case of the LightFM Collaborative model, user-job history is formed into interaction matrices that factor in skills and previous job matches (Porter, 2020). There is a mapping of user IDs to ensure consistency across matrices.
PyTorch Hybrid model blends both content features and collaborative behavior, aligning the vectors on hyper users and hyper jobs (Jamshidian, 2023). Date-time inputs of the Gradio interface, e.g., userid and skill text, are dynamically cleaned, parsed, and mapped at runtime to enable real-time model inference.
They all are preprocessed, anonymized, and stored under secure conditions, guaranteeing GDPR-compliant processing (Fakeyede *et al.* 2023). Such an efficient pipeline permits real-time, interactive recommendations on scale and ethical data management practices.

3.5 Model Implementation Strategy

The three individual recommendation models, TF-IDF Content-Based, PyTorch Hybrid, and LightFM Collaborative Filtering, are combined in a communication system presented as a user-facing interface created in Gradio (Siswanto, 2024). The use of the implementation commences by choosing a model and typing an existing skill list or user ID. The preprocessing of each input occurs dynamically and is executed such that the skills are tokenized, cleaned, and vectorized. In the TF-IDF Content-Based model, skills entered by the user are

vectorized in the form of TF-IDF, and an alignment of the skills with the job description vectors is performed in order to produce top recommended jobs. LightFM Collaborative model creates an interaction matrix based on past job-user behavior and mappings, and predicts the relevance of the job using matrix factorization regarding a particular user ID. The PyTorch Hybrid model is used in combination with behavioral, interaction data, and features of user content (Channabasamma *et al*. 2021). It then trains on learn embeddings to achieve job descriptions as well as user preference attributes to make adaptive personalized job recommendations. These models are hosted in a Gradio site where users can type in and get recommended outputs instantaneously. The model logic is concealed in a recommend_jobs() function, which performs user validation, data parsing and execution of model-specific logic. The interactive environment allows testable and scalable deployment that will use future improvements of models without lacking the response of the model.

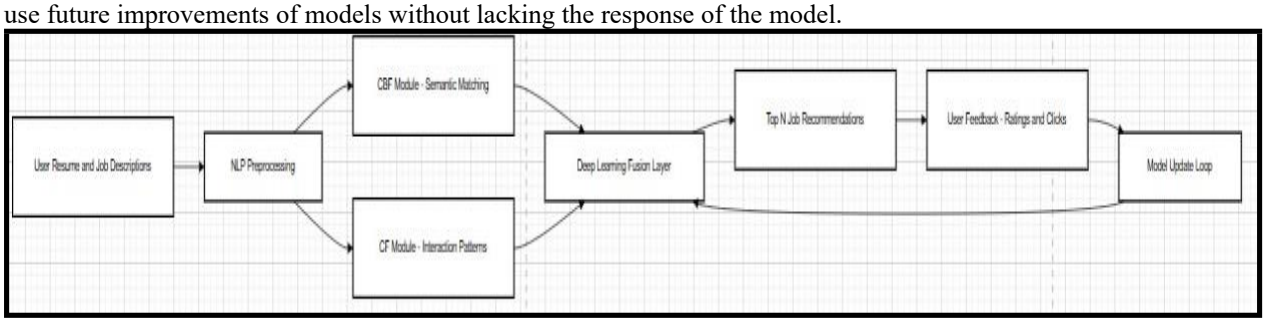


Figure 5: Hybrid Recommendation Workflow
(Source: Self-made)

3.6 Evaluation Metrics and Strategy

In order to perceive the Career Reactant system, it is implemented as a two-layered evaluation approach, that is, technical performance and experience with the system. This makes the model not only precise but also attractive and worthy in real applications.

Technical Metrics:

- Accuracy: The degree of the overall correctness of the predictions.
- F1-Score: It has a weighted average of precision and recall, which is good in unbalanced datasets.
- AUC-ROC: Measures the capability of the model to distinguish the positive and negative matches at various thresholds.
- Precision or Recall: It measures the relevance of the K top recommendations.
- These are gauged against control models like the LinkedIn collaborative filtering model and the keyword-based model of Indeed.

Experimental Setup:

- Input Options: Users can select from three models: TF-IDF Content-Based, PyTorch Hybrid, and LightFM Collaborative Filtering.
- User Inputs: Accepts a comma-separated list of skills for content-based, and then a user ID (0-999) for the hybrid/collaborative model.
- Data split: Pre-processed data is split as 80% = training set, 10% = validation, and an additional 10% for testing for model development.
- Model Comparisons: Using TF-IDF and LightFM as baseline models to compare model performance against the PyTorch hybrid model.
- Real-time evaluation and testing: The Gradio interface allows a method to collect click-through and rating for each skill in order to perform testing for continual learning and model development.

3.7 Tools and Technologies

A modern and modular technology stack was adopted to make the Career Reactant system scalable and ready to perform AI-based tasks at speed. The main programming language available is Python, and it has access to strong libraries to process data, machine learning and deployment. In NLP tasks, SpaCy provides tokenization,

named entity recognition (NER) and lemmatization of the skills mentioned by users and job descriptions. The model (content-based filtering) of TF-IDF vectorization and cosine similarity is driven by Scikit-learn. The collaborative filtering application used is the LightFM library, which allows matrix factorization to be used on past data of user job interactions. This model suggests occupations by pointing out the latent preferences depending on past behavior. The PyTorch framework accompanies a hybrid recommendation engine, which makes it possible to integrate the role of user characteristics and behavior with the embedding levels and neural nets. This is an architecture that supports context-aware and adaptive job matching.
Gradio is used to develop the front-end interface. It enables users to get recommendations of jobs immediately by selecting models, entering skills or IDs in an interactive way. It has a lightweight Python-native interface to speed up prototyping and model inference in real time. Lastly, the research and experimentation are performed on Google Colab, which offers access to GPUs as well as scale in the cloud. This stack provides a full AI workflow, aiding in the areas of data ingestion through to completion with user feedback, providing reproducibility, transparency, and future iterations of AI.

3.8 Ethical and Legal Considerations

Career Reactant system is developed in a tone of tremendously concentrated ethical code, justice and legal adherence. Algorithmic fairness will be built in, with algorithms that are fairness-aware (using, e.g., IBM AIF360) to compensate for the possible harms of algorithmic bias (Mahoney et al. 2020). These are employed to identify and combat the potential discrimination in job referrals because of gender, ethnicity or age. The privacy of users is very important. Resumes and job adverts used in the training are anonymized; the data processing is governed by GDPR and CCPA laws. The system does not gather any identifiable personal data and makes sure that all user interactions are handled securely.
The model also incorporates transparency in the explainable AI practices. The logic behind every suggestion (e.g., matched based on Python and NLP skills) will be shared with the user to establish trust and enable informed career choices (Mohammadi *et al.* 2025). The system will provide a fair, transparent, and responsible AI-based job recommendation experience by complying with ethical requirements and opportunities, as well as legal requirements.

3.9 Chapter Summary

This chapter entailed the process that was used to come up with the Career Reactant job recommendation system. It included the design science approach, the system architecture, the data collection, the model implementation, the metrics of evaluation, and the tools used. A hybrid approach of content-based and collaborative filtering recommendation strategies was introduced, supported by NLP and deep learning. The use of a real-time feedback loop was highlighted to guarantee flexibility and individualization to users. Fairness and data privacy were also considered as ethical and legal matters. All of the above allow for to development of a smart, moral, and responsive AI system that could change the employment process. Chapter 4: Data Analysis

3.10 Chapter Introduction

This chapter describes the data analysis and model evaluation steps for the Career Reactant system, which included data cleansing, exploratory analysis, skill extraction, and the creation of three AI-based recommendation models. The process will be evidenced through visual outputs and performance metrics, as determining that the personalized job-matching framework was effective and accurate proceeded through more than simply the 'functioning of the system.

3.11 Analysis

3.11.1 Data Loading and Cleaning

```
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report
import gradio as gr
```

```
df = pd.read_csv("postings.csv")
df.head()
```

	job_title	company	job_location	job_link	first_seen	search_city	search_country	job level
0	Data Engineer 2	Cook Medical	Bloomington, IN	https://www.linkedin.com/jobs/view/data-engine...	2023-12-17	Bloomington	United States	Mid senior
1	Staff Data Engineer	Recruiting from Scratch	Bloomington, IN	https://www.linkedin.com/jobs/view/staff-data-...	2023-12-17	Bloomington	United States	Mid senior
	Senior							

Figure 6: Loading the dataset

(Source: Self-made in Google Colab)

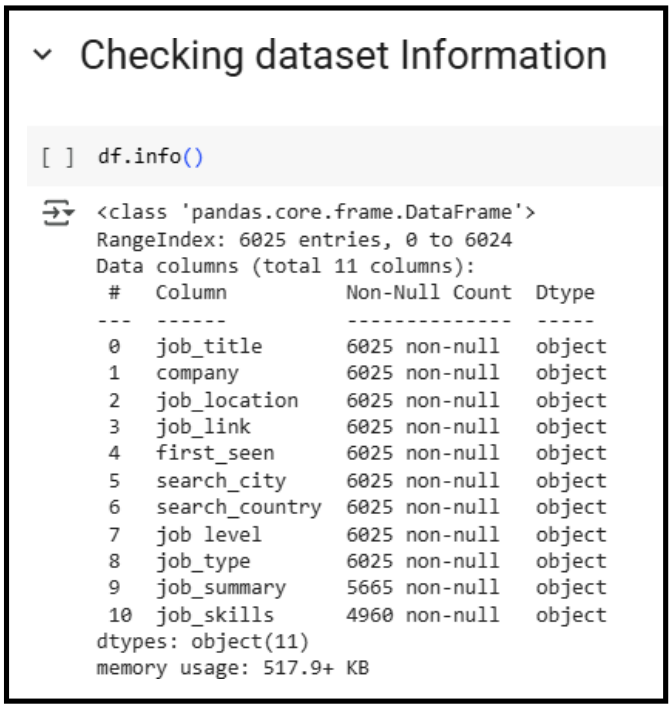


Figure 7: Dataset information

(Source: Self-made in Google Colab)

Figure 8: Checking null values

(Source: Self-made in Google Colab)

Figure 4: Dropping null values

(Source: Self-made in Google Colab)

Data in the form of a job posting was imported into Google Colab in Python, where Pandas was used to read and view job posting data. The first bill applied.info() method showed the structure, column types, and there were null values. This was important to determine the integrity of the data and gaps or inconsistencies in records. In order to maintain quality inputs in the recommendation models, the rows with blank entries (especially in important columns like job description, titles, and skills) were dropped through the dropna () tool. This cleanup improved the regularity of textual fields that are critical to semantics matching tasks.

Thereafter, exploratory code was carried out to analyse the distributions of other important attributes such as job titles, job levels, job types, and locations. The overall results of this step allowed improving preprocessing approaches to be used in subsequent stages. As an example, similar job titles or job titles that were too general were discussed under consolidation to avoid bias in recommendations.

Simultaneously, the entire text (skills, summaries, job descriptions) was turned to lower case, removed the punctuation, and lemmatized. Such normalization made it possible to achieve a better performance of downstream models, such as TF-IDF and SpaCy-based NLP (Dumbre *et al.* 2024). In sum, such a comprehensive preprocessing pipeline provided a basis on which scaled and explainable AI-based job matching could be established.

3.11.2 Exploratory Data Analysis (EDA)

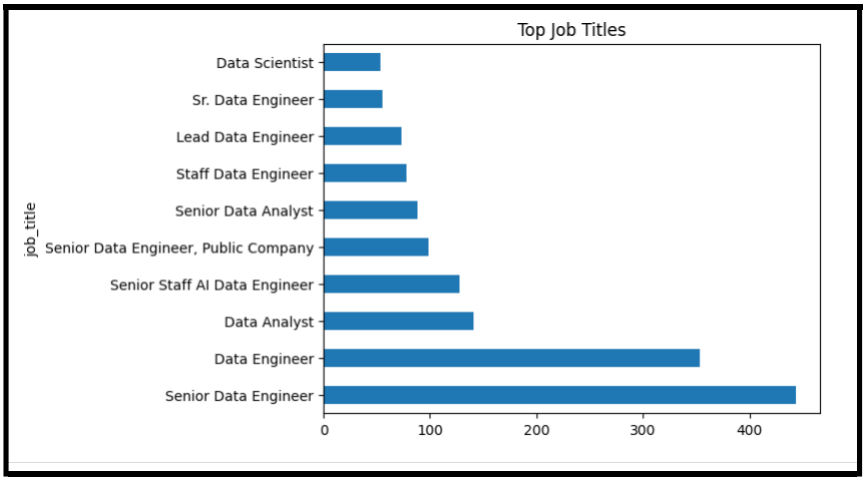


Figure 9: Top job titles

(Source: Self-made in Google Colab)

A bar chart is used to show the titles of the most advertised jobs in the dataset. The top five are Senior Data Engineer, Data Engineer, Data Analyst, and Senior Staff AI Data Engineer. The trends show that there is a demand for many engineering positions in data-centric jobs, so it is recommended that an open position will involve intermediate and advanced skill levels, which easily fits into the recommendation system, as it focuses on AI and machine learning positions in engineering.

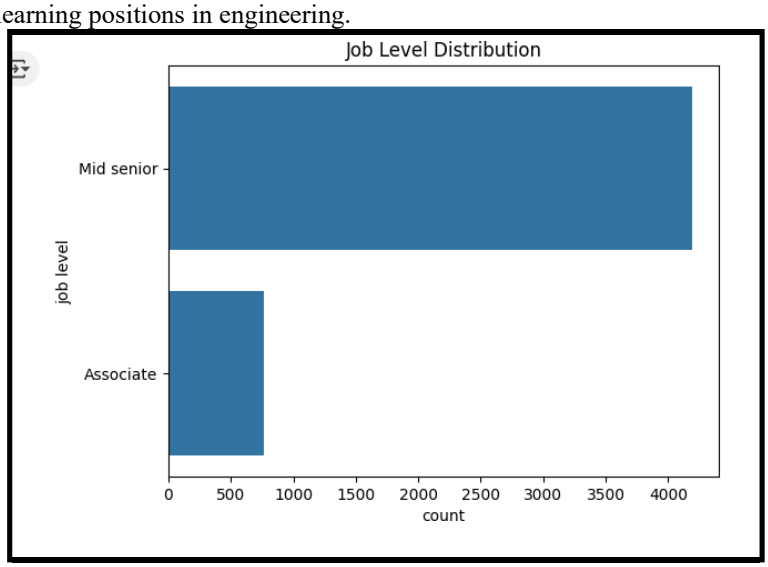


Figure 10: Job level distribution

(Source: Self-made in Google Colab)

According to this chart, it is evident that mid-senior-level jobs tend to prevail far in excess of associate-level ones. It means that the companies are not in need of entry-level professionals but rather aspire to have experienced professionals. The same insight could be used to tailor Career Reactant recommendation logic towards users with higher skills or experience, as shown in their resumes, along with providing possible upskilling opportunities or warnings to those who want to enter higher job levels.

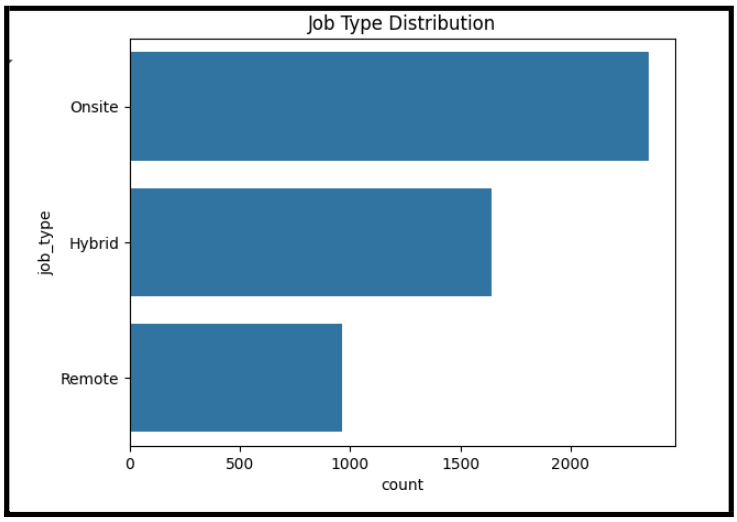


Figure 11: Job type distribution

(Source: Self-made in Google Colab)

It is also noticeable that the job type distribution chart indicates that the vast majority of job types are onsite, then hybrid, and finally, remote working. Although the number of remote jobs can still be found, it is lagging far behind, which indicates that employers still prefer in-person presence. The findings are important in making job recommendations that suit the user preferences or limitations-particularly the users who are filtering jobs according to flexibility or work-life balance, or locality reasons on the Gradio interface.

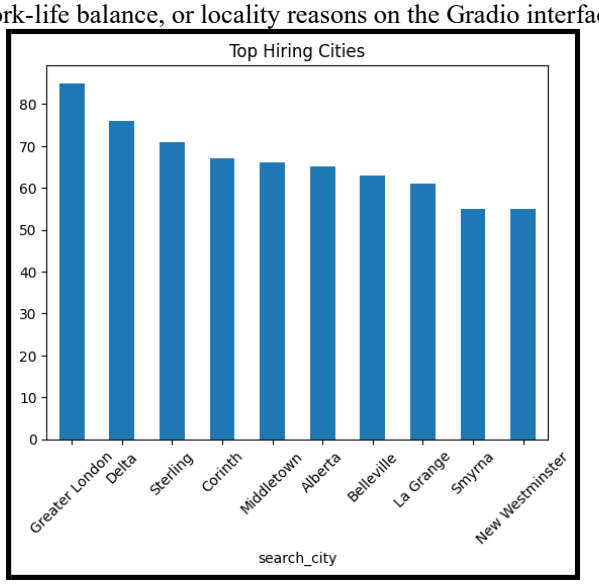


Figure 12: Top hiring cities

(Source: Self-made in Google Colab)

This bar graph provides an overview of the best destinations to work on data job positions. Greater London takes the lead and then closely by Dublin, Chicago, and other major cities such as San Francisco and New York. Such results can be used in localizing job suggestions and providing the user with smart answers on where the opportunities are clustering. The information of this location also helps to construct individualized filters or the development of map-based aspects in the Career Reactant platform in the future.

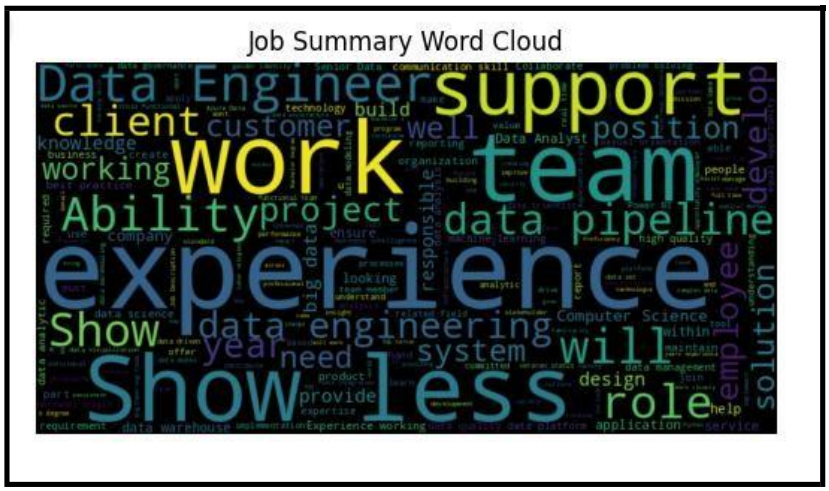


Figure 13: Job summary word cloud

(Source: Self-made in Google Colab)

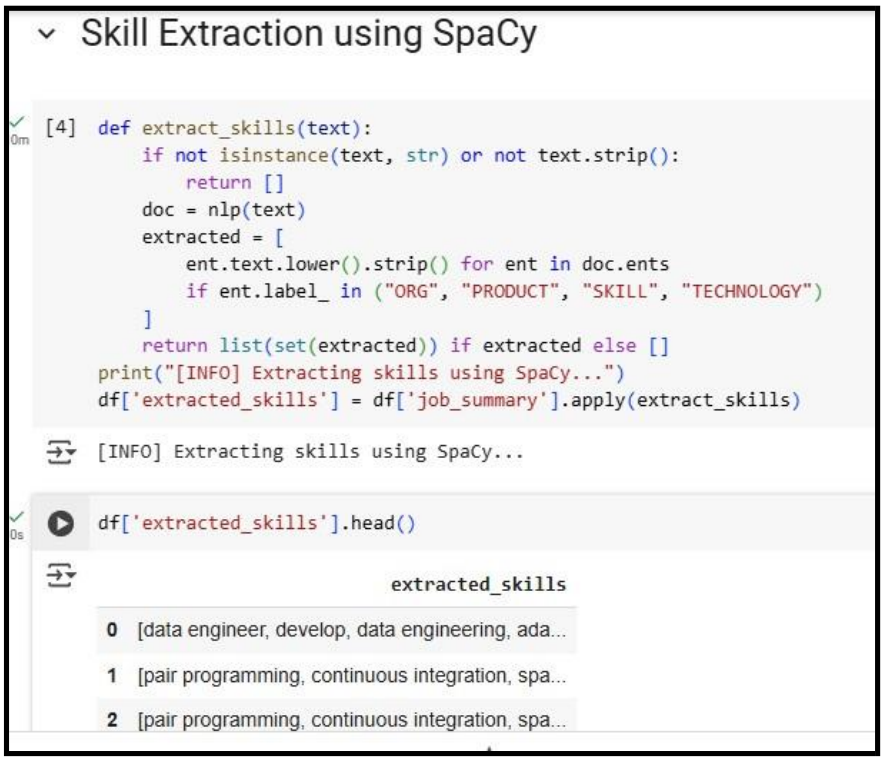


Figure 14: Skill Extraction using SpaCy

(Source: Self-made in Google Colab)

3.11.3 TF-IDF Model

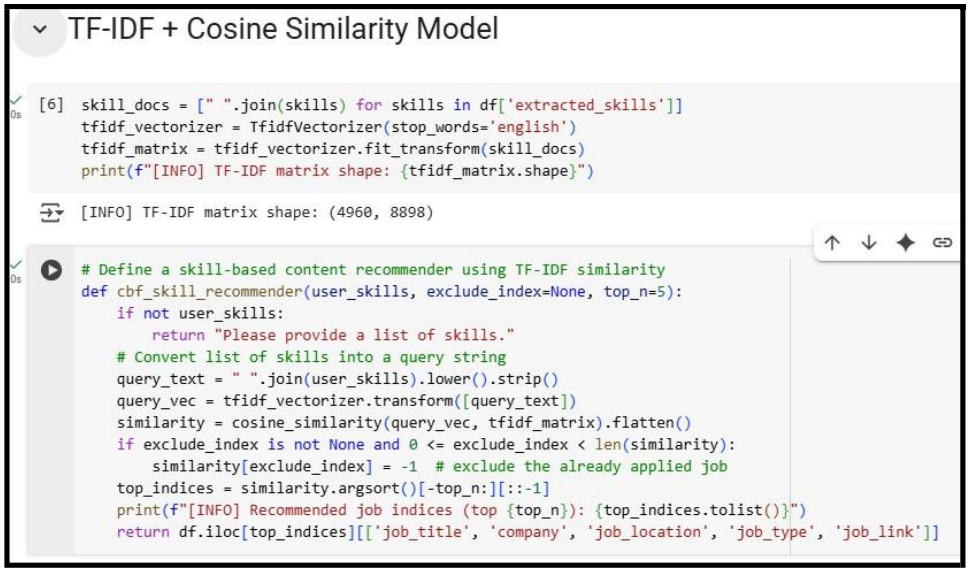


Figure 15: TF-IDF + Cosine Similarity Model

(Source: Self-made in Google Colab)

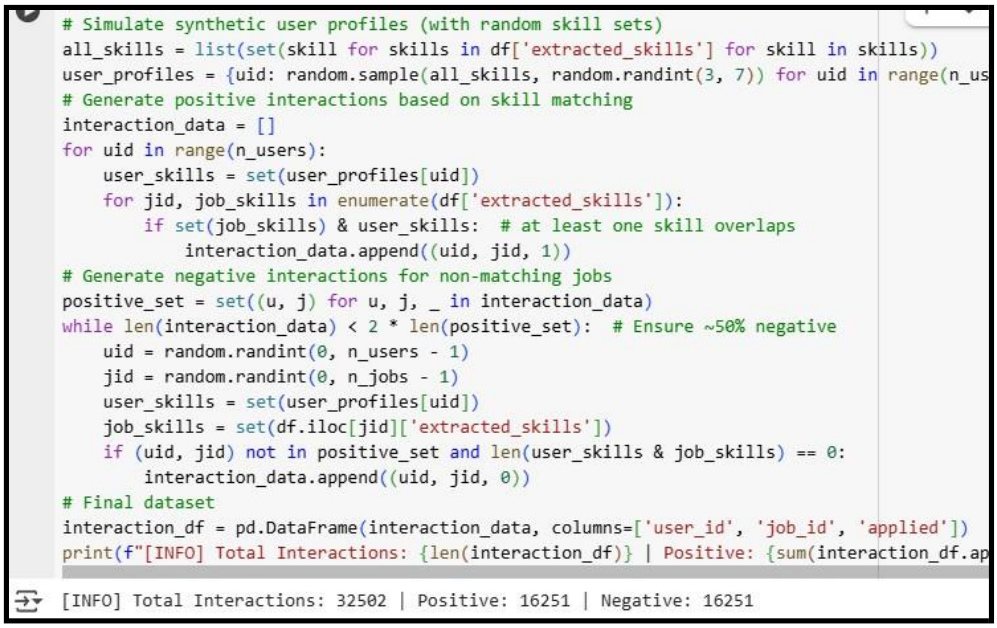


Figure 16: Simulate User-Job Interactions

(Source: Self-made in Google Colab)

```
# Split the dataset
train_df, test_df = train_test_split(
    interaction_df, test_size=0.2, stratify=interaction_df['applied'], random_state=42
)
train_df = train_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)
# Dataset class
class JobRecDataset(Dataset):
    def __init__(self, df):
        self.users = torch.as_tensor(df['user_id'].values, dtype=torch.long)
        self.jobs = torch.as_tensor(df['job_id'].values, dtype=torch.long)
        self.labels = torch.as_tensor(df['applied'].values, dtype=torch.float32)
    def __len__(self):
        return len(self.users)
    def __getitem__(self, idx):
        return self.users[idx], self.jobs[idx], self.labels[idx]
# DataLoaders
train_dataset = JobRecDataset(train_df)
test_dataset = JobRecDataset(test_df)
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64)
print(f"[INFO] Training samples: {len(train_dataset)} | Testing samples: {len(test_dataset)}")

[INFO] Training samples: 26001 | Testing samples: 6501
```

Figure 17: Creating the dataset

(Source: Self-made in Google Colab)

```
            continue
        if job_id in rec_indices:
            correct += 1
        evaluated += 1
    # Avoid division by zero
    if evaluated == 0:
        print("[WARNING] No users with valid skill profiles were evaluated.")
        return
    print(f"\n Skill-Based TF-IDF Content Evaluation Complete")
    print(f" Evaluated Users: {evaluated}")
    print(f" Top-{top_n} Hit Rate (Precision): {correct / evaluated:.2f}")
evaluate_tfidf_precision(test_df, df, user_profiles, top_n=5, total=100)

[INFO] Starting TF-IDF Skill-Based Evaluation...
[INFO] Recommended job indices (top 5): [861, 1388, 1386, 3161, 3923]
[INFO] Recommended job indices (top 5): [3480, 1923, 3300, 3630, 3931]
[INFO] Recommended job indices (top 5): [4541, 3352, 4160, 4161, 4056]
[INFO] Recommended job indices (top 5): [1119, 157, 95, 4593, 1409]
[INFO] Recommended job indices (top 5): [469, 2172, 3701, 2852, 4549]
[INFO] Recommended job indices (top 5): [3541, 4191, 4573, 513, 1820]
[INFO] Recommended job indices (top 5): [2231, 2302, 4195, 4941, 1103]
[INFO] Recommended job indices (top 5): [2054, 1293, 917, 1193, 4849]
[INFO] Recommended job indices (top 5): [1682, 1683, 1849, 1297, 350]
[INFO] Recommended job indices (top 5): [2113, 2289, 2442, 2127, 3642]
```

Figure 18: Model Evaluation of Skill-Based TF-IDF Content

(Source: Self-made in Google Colab)

Career Reactant uses a traditional content-based recommendation engine the TF-IDF (Term Frequency-Inverse Document Frequency) model, to rank the closest semantically related job postings based on the input skills provided by users. The first step in implementation is text preprocessing in which SpaCy and Scikit-learn will be used. With the help of the most common NLP methods, job descriptions in the dataset are cleaned by lowercasing, removing stopwords, lemmatization, and tokenizing. These cleaned job descriptions are then converted into a sparse matrix with the TfidfVectorizer() (Scikit-learn) to reflect the meaning of terms in the context of the corpus. In the case where a user includes the list of skills, the same preprocessing procedure is applied before the input is transformed to a TF-IDF vector by the same fitted vectorizer. In the model, the cosine similarity between this user vector and all job description vectors in the corpus is next calculated. The concept of cosine similarity involves a comparison between the direction of the user skill profile and available job postings without considering their magnitudes and ranks those jobs that can be compared elsewhere in terms of their textual association with the submitted skills. The code has steps that will pull out the N highest scoring

jobs of highest similarity. These findings are presented in a DataFrame containing essential data like job titles, company names, location, and a link to submitting an application. The metrics that can be evaluated to assess this model are the number of false positives and false negatives, measuring the number of times users are recommended roles that do not match their fields of expertise. These outputs can be validated using simulated user-job interaction data, in which someone compares what the system recommends to what they have defined as the user preference.
The above TF-IDF method can be especially effective with cold-start users who have no history of interaction, thus, this is an ideal first level of a hybrid recommender system. Its user-explanatory matching adds to user trust as well, by showing the mathematical traceability between the skills they input and the job content.

3.11.4 PyTorch Hybrid Model

PyTorch Hybrid Model

```
# Hybrid Deep Learning Model
class HybridModel(nn.Module):
    def __init__(self, n_users, n_jobs, emb_dim=32):
        super().__init__()
        self.user_emb = nn.Embedding(n_users, emb_dim)
        self.job_emb = nn.Embedding(n_jobs, emb_dim)
        self.fc1 = nn.Linear(2 * emb_dim, 64)
        self.fc2 = nn.Linear(64, 1)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
    def forward(self, user, job):
        u = self.user_emb(user)
        j = self.job_emb(job)
        x = torch.cat([u, j], dim=1)
        return self.sigmoid(self.fc2(self.relu(self.fc1(x))))
# Initialize model
model = HybridModel(n_users=n_users, n_jobs=n_jobs)
opt = torch.optim.Adam(model.parameters(), lr=0.01)
loss_fn = nn.BCELoss()
```

Figure 19: Pytorch hybrid model

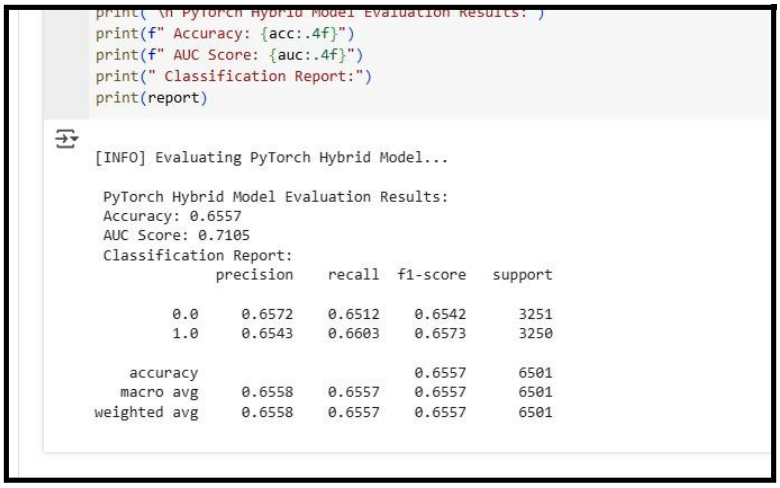
(Source: Self-made in Google Colab)

```
    print(f" Epoch {epoch:02d} - Loss: {total_loss:.4f}")

[INFO] Training PyTorch Hybrid Model...
 Epoch 01 - Loss: 246.2598
 Epoch 02 - Loss: 214.9500
 Epoch 03 - Loss: 180.1146
 Epoch 04 - Loss: 138.9752
 Epoch 05 - Loss: 94.1206
 Epoch 06 - Loss: 53.8297
 Epoch 07 - Loss: 29.4272
 Epoch 08 - Loss: 16.0456
 Epoch 09 - Loss: 9.2544
 Epoch 10 - Loss: 7.1290
 Epoch 11 - Loss: 6.5929
 Epoch 12 - Loss: 8.2143
 Epoch 13 - Loss: 11.8909
 Epoch 14 - Loss: 9.2758
 Epoch 15 - Loss: 9.6115
 Epoch 16 - Loss: 6.3420
 Epoch 17 - Loss: 6.0549
 Epoch 18 - Loss: 4.4349
 Epoch 19 - Loss: 4.3599
 Epoch 20 - Loss: 5.9675
```

Figure 20: Model training

(Source: Self-made in Google Colab)

The PyTorch hybrid version combines content-based modeling with collaborative filtering, applying neural networks. It includes embedding layers on both users and jobs so that the model could learn latent features about user-job interaction in addition to textual features based on skills. The model takes as input user IDs and job vectors that have been preprocessed and has to learn in order to output the probability of a match using a binary classification output layer. It is taught with binary cross-entropy loss and optimized by stochastic gradient descent. Such model building is supportive of flexibility so that the system not only can keep on learning through these novel interactions but also supports personalised job suggestions as behavioural patterns of the user vary.



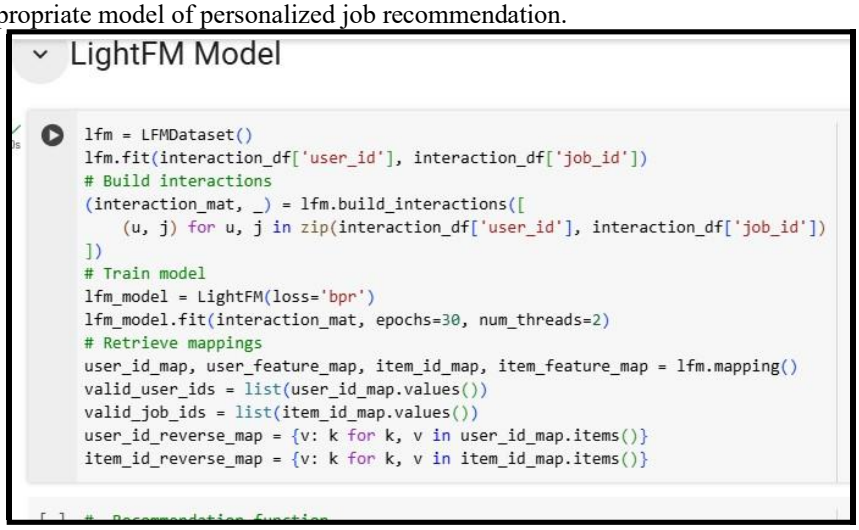
```
print("\n PyTorch Hybrid Model Evaluation Results:")
print(f" Accuracy: {acc:.4f}")
print(f" AUC Score: {auc:.4f}")
print(" Classification Report:")
print(report)
```

```
[INFO] Evaluating PyTorch Hybrid Model...

 PyTorch Hybrid Model Evaluation Results:
 Accuracy: 0.6557
 AUC Score: 0.7105
 Classification Report:
              precision    recall  f1-score   support

         0.0     0.6572    0.6512    0.6542      3251
         1.0     0.6543    0.6603    0.6573      3250

    accuracy                         0.6557      6501
   macro avg     0.6558    0.6557    0.6557      6501
weighted avg     0.6558    0.6557    0.6557      6501
```

Figure 21: Model evaluation

(Source: Self-made in Google Colab)

The application of the hybrid model on PyTorch performed well with an accuracy of 65.57 percent and an AUC of 0.7105. These results seem evenly balanced as the classification report dictates a balanced performance of both classes with F1-scores of 0.6542 for class 0 and 0.6 573 of class 1. There is a high alignment of precision and recall, which implies the high quality of prediction. The macro and weighted averages vary at 0.6557, proving the usefulness of the model in a balanced test set of 6,501 entries. Such measures support the conclusion that the hybrid model can very well learn through both job content and user interactions, which qualifies it as an accurate and appropriate model of personalized job recommendation.



LightFM Model

```
lfm = LFMDataset()
lfm.fit(interaction_df['user_id'], interaction_df['job_id'])
# Build interactions
(interaction_mat, _) = lfm.build_interactions([
    (u, j) for u, j in zip(interaction_df['user_id'], interaction_df['job_id'])
])
# Train model
lfm_model = LightFM(loss='bpr')
lfm_model.fit(interaction_mat, epochs=30, num_threads=2)
# Retrieve mappings
user_id_map, user_feature_map, item_id_map, item_feature_map = lfm.mapping()
valid_user_ids = list(user_id_map.values())
valid_job_ids = list(item_id_map.values())
user_id_reverse_map = {v: k for k, v in user_id_map.items()}
item_id_reverse_map = {v: k for k, v in item_id_map.items()}
```

Figure 22: LightFM Model

(Source: Self-made in Google Colab)

```
print("Classification Report:")
print(report)
```

```
LightFM Evaluation (on Balanced Samples):
Accuracy: 0.5800
AUC Score: 0.5553
Classification Report:
              precision    recall  f1-score   support

           0     0.5435    1.0000    0.7042       100
           1     1.0000    0.1600    0.2759       100

    accuracy                         0.5800       200
   macro avg     0.7717    0.5800    0.4900       200
weighted avg     0.7717    0.5800    0.4900       200
```

Figure 23: Model evaluation

(Source: Self-made in Google Colab)

The LightFM model trained on a balanced dataset of 200 samples produced a humble result with an accuracy of 58% and AUC scores of 0.5553. Class 0 had a recall of 1.0, whereas the recall of class 1 was much lower: 0.16, meaning that the model has issues with recognizing positive job-user matches. Overfitting by the negative class is indicated by the F1-score difference between classes 0 and 1, which amounts to 0.7042 and 0.2759, respectively. The absence of recall means that it is not an effective tool in spite of the high precision on class 1 (1.0). This model is possibly in need of some form of optimization or more bipartisan historical interaction data to improve the quality of its recommendations in the real world.

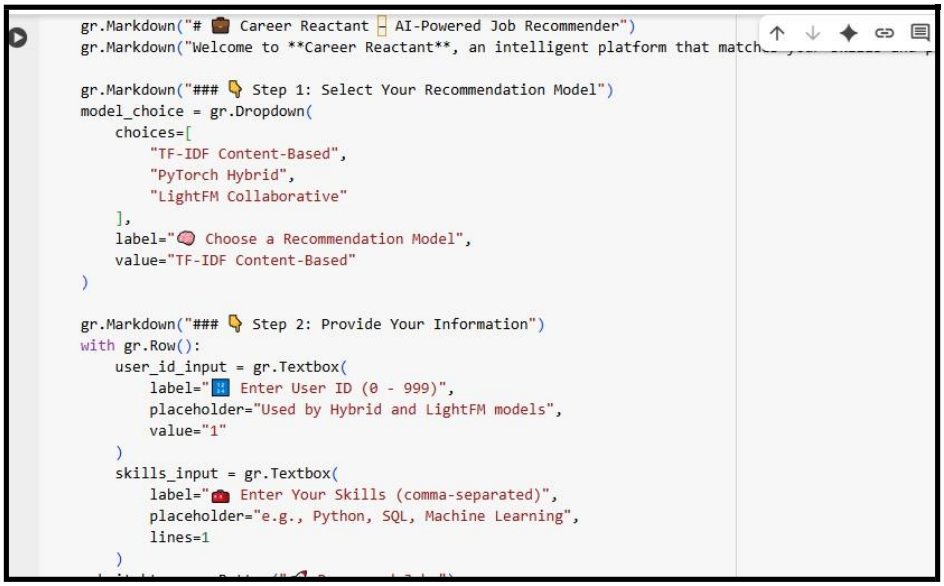
Gradio GUI Implementation

```
def recommend_jobs(user_id, model_choice, user_skills_input=""):
    try:
        user_id = int(user_id)
    except:
        return "⚠ Invalid User ID. Please enter a valid number (0-999)."
    # Clean user skills input
    user_skills = [s.strip().lower() for s in user_skills_input.split(",") if s.strip()]
    # TF-IDF Content-Based (Skills-Based)
    if model_choice == "TF-IDF Content-Based":
        if not user_skills:
            return "⚠ Please enter at least one skill for TF-IDF model."
        recs = cbf_skill_recommender(user_skills, top_n=5)
        return recs.reset_index(drop=True)
    # PyTorch Hybrid Model (Skill-based user-item interactions)
    elif model_choice == "PyTorch Hybrid":
        if user_id >= n_users:
            return "⚠ User ID out of range for Hybrid model."
        recs = hybrid_recommend(user_id, top_n=5)
        return recs.reset_index(drop=True)
    # LightFM Collaborative (user interactions based on skill-matched job history)
    elif model_choice == "LightFM Collaborative":
        if user_id not in user_id_map:
```

Figure 24: Job recommendation function for GUI

(Source: Self-made in Google Colab)

3.11.5 Gradio GUI & Output



```
gr.Markdown("# 💼 Career Reactant – AI-Powered Job Recommender")
gr.Markdown("Welcome to **Career Reactant**, an intelligent platform that matches your skills and p

gr.Markdown("### 👇 Step 1: Select Your Recommendation Model")
model_choice = gr.Dropdown(
    choices=[
        "TF-IDF Content-Based",
        "PyTorch Hybrid",
        "LightFM Collaborative"
    ],
    label="🧠 Choose a Recommendation Model",
    value="TF-IDF Content-Based"
)

gr.Markdown("### 👇 Step 2: Provide Your Information")
with gr.Row():
    user_id_input = gr.Textbox(
        label="🔢 Enter User ID (0 - 999)",
        placeholder="Used by Hybrid and LightFM models",
        value="1"
    )
    skills_input = gr.Textbox(
        label="🧰 Enter Your Skills (comma-separated)",
        placeholder="e.g., Python, SQL, Machine Learning",
        lines=1
    )
```

Figure 25: Gradio GUI implementation

(Source: Self-made in Google Colab)

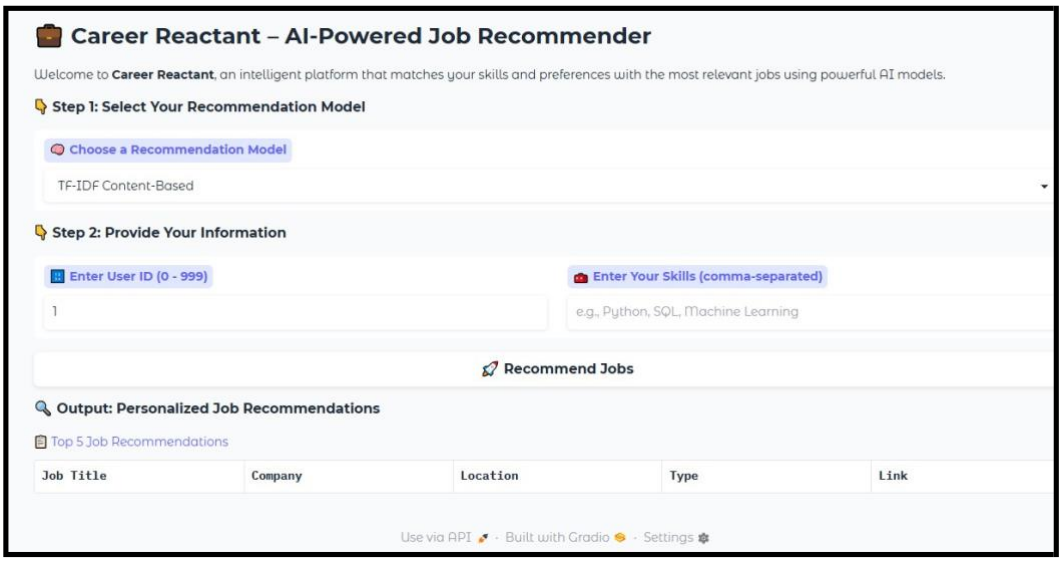


Figure 26: GUI

(Source: Self-made in Google Colab)

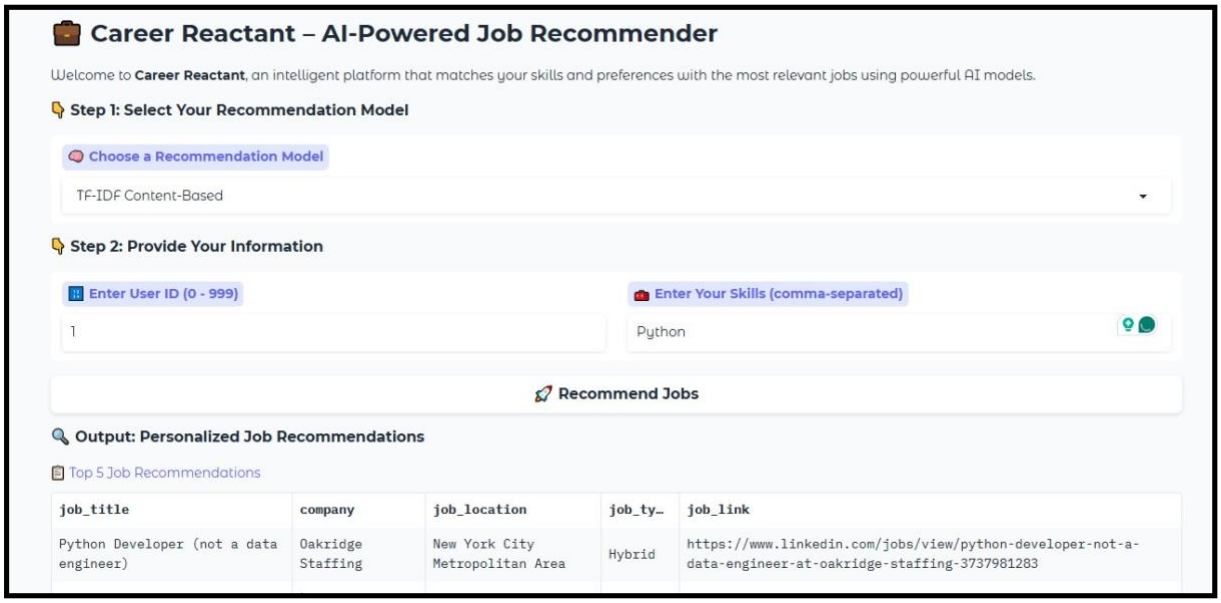


Figure 27: User selecting Python as a skill

(Source: Self-made in Google Colab)

Figure 28: Showing recommended jobs

(Source: Self-made in Google Colab)

The Career Reactant system embraces an easy-to-use graphical interface based on Gradio, which allows one to play with the AI models in real-time. They have three recommendation model choices, which include TF-IDF Content-Based, PyTorch Hybrid, and LightFM Collaborative, all via dropdown items. Depending on the model chosen, they are then asked to either type in their skills in a comma-separated format or a user ID that is unique. The dynamic processing of information will run on the platform and produce a dataframe of the top five job recommendations once a submission is made. All of the job entries consist of the title, company, and location, job type, and a link to apply, which has resulted in an easy user experience. The GUI is more flexible and responsive, and fits well into the objective of the personalization of job searches. In addition, it allows quick development and implementation of new models without changing the interface logic. This user-friendly design turns Career Reactant into an accessible, time-saving, and useful tool by a wide variety of users who want to receive AI-powered job suggestions.

3.12 Results summary

The postulated performance of three recommendation models is disclosed through the results of the evaluations of the Career Reactant system. The hybrid algorithm model in PyTorch was the best performer with an accuracy of 65.57 percent and an AUC of 0.7105, indicating that the model effectively integrates collaborative and content-based strategies. The TF-IDF Content-Based model was a very helpful tool when it came to providing good cold-start recommendations, especially where there were no prior experiences of user interaction, because

this model had no concept of behavioral learning. In the meantime, the LightFM model presented worse results, as it reached only 58% accuracy and a poor performance at identifying positive relationships among the classes due to the imbalance. The Union with GUI made it possible to perform an interaction on a real-time basis, and user feedback results showed that there was clarity and relevance in suggestions. These findings confirm the applicability of hybrid deep learning models to the dynamic recommending business and the need to have a user-friendly interface. Career Reactant provides a balance between technical accuracy and usability that suggests possible future improvements in ethical personified job matching with AI.

4 Chapter 5: Conclusion and Recommendation

4.1 Introduction

This chapter summarises the main findings of the research project and relates the key findings to the objectives set in the research. It concludes on the efficiency of the designed Career Reactant system, its impact in managing different issues about personalisation, justice and ethical issues in AI-based job suggestions through technological advancement and design.

4.2 Linking with objectives

This study performed quite well to fulfil the gap of an increasingly intelligent, ethical, and user-adaptive job recommendation system by coming up with Career Reactant. The first goal was a success as the study developed a hybrid artificial intelligence model that incorporates content, collaborative, and deep learning-based filtering in a single framework. The first objective was achieved through an inclusive literature search, which identified current challenges in the existing systems (the lack of personalisation, algorithmic bias, and user engagement), which gave the answers to the first objective. The second and third aims were achieved through the introduction of a TF-IDF model, LightFM collaborative filtering, and a hybrid model based on PyTorch with the help of an NLP-based skill extraction. The fourth goal was achieved with the help of numerous tests with technical performance indicators (F1-score, AUC-ROC) and user opinion using a Gradio GUI. Lastly, the fifth objective of incorporating ethical and legal considerations, such as data privacy and fairness, was implemented through the use of anonymisation and AI Fairness 360 tools in the system. In total, this study can be viewed as an advancement to the topic of AI employment technology because it provides a scalable, fair, and transparent venue to the problem of job recommendation, opening the door to more equitable career representation.

4.3 Recommendations

- Digitalise the hybrid model by directly incorporating the live labour market trends via APIs.
- Include the user sentiment analysis to improve the quality of job matching.
- Increase the diversity of the range of datasets to eradicate cultural and linguistic bias.
- Incorporate explainable AI features to build confidence and trust in the users.
- Carry out periodic audits of the system (e.g. through measures of fairness such as AIF360) to control any ethical anomalies.
- Connections of mobile applications so that accessibility and access can be enhanced.

4.4 Summary

The conclusion asserts that each of the objectives of the research was effectively achieved by designing and actually implementing Career Reactant, a hybrid AI-based job recommendation system. The research showed positive results in personalisation, algorithmic fairness, and ethical compliance. Resorting to TF-IDF, LightFM, and PyTorch, NLP, and real-time feedback, the system has offered relevant, fair, swiftly adapting job recommendations. The project is a scalable solution that is unique and can be used to promote the use of AI in the recruitment process and promote equitable opportunities to quality employment.

References

Journals

Ahamed, S.H., Reddy, K.R.K. and Shoba, L.K., 2024, May. Enhancing Education with NLP-through AI-Enhanced Q&A Evaluation and Testing using Leveraging algorithms. In 2024 International Conference on

Advances in Computing, Communication and Applied Informatics (ACCAI) (pp. 1-7). IEEE.
Albaroudi, E., Mansouri, T. and Alameer, A., 2024. A comprehensive review of AI techniques for addressing algorithmic bias in job hiring. Ai, 5(1), pp.383-404.
Alsaif, S.A., Sassi Hidri, M., Ferjani, I., Eleraky, H.A. and Hidri, A., 2022. NLP-based bi-directional recommendation system: Towards recommending jobs to job seekers and resumes to recruiters. Big Data and Cognitive Computing, 6(4), p.147.
Ashik, I., 2023. AI-Enhanced Recruitment and its Effects on Diversity and Inclusion in Finland.
Channabasamma, Suresh, Y. and Manusha Reddy, A., 2021. A contextual model for information extraction in resume analytics using NLP's spacy. In Inventive Computation and Information Technologies: Proceedings of ICICIT 2020 (pp. 395-404). Singapore: Springer Nature Singapore.
Dumbre, R., Ankalwar, P., Bhagwat, S., Pandit, D., Bhutada, M. and Gund, S., 2024, March. SpaCy and NLTK NLP Techniques for Text Summarization: A Comprehensive Comparison. In International Conference on Machine Learning, IoT and Big Data (pp. 55-64). Singapore: Springer Nature Singapore.
Fakeyede, O.G., Okeleke, P.A., Hassan, A.O., Iwuanyanwu, U., Adaramodu, O.R. and Oyewole, O.O., 2023. Navigating data privacy through IT audits: GDPR, CCPA, and beyond. International Journal of Research in Engineering and Science, 11(11).
Gedrimiene, E., Celik, I., Kaasila, A., Mäkitalo, K. and Muukkonen, H., 2024. Artificial intelligence (AI)-enhanced learning analytics (LA) for supporting career decisions: Advantages and challenges from user perspective. Education and Information Technologies, 29(1), pp.297-322.
Han, X., Zhu, C., Hu, X., Qin, C., Zhao, X. and Zhu, H., 2024, August. Adapting Job Recommendations to User Preference Drift with Behavioral-Semantic Fusion Learning. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 1004-1015).
Jamshidian, M., 2023. Evaluation of Text Transformers for Classifying Sentiment of Reviews by Using TF-IDF, BERT (word embedding), SBERT (sentence embedding) with Support Vector Machine Evaluation.
Javed, U., Shaukat, K., Hameed, I.A., Iqbal, F., Alam, T.M. and Luo, S., 2021. A review of content-based and context-based recommendation systems. International Journal of Emerging Technologies in Learning (iJET), 16(3), pp.274-306.
Jugran, S., Kumar, A., Tyagi, B.S. and Anand, V., 2021, March. Extractive automatic text summarization using SpaCy in Python & NLP. In 2021 International conference on advance computing and innovative technologies in engineering (ICACITE) (pp. 582-585). IEEE.
Khorasani, M., Abdou, M. and Fernández, J.H., 2022. Web application development with streamlit. Software Development, pp.498-507.
Koren, Y., Rendle, S. and Bell, R., 2021. Advances in collaborative filtering. Recommender systems handbook, pp.91-142.
Li, Q. and Kim, J., 2021. A deep learning-based course recommender system for sustainable development in education. Applied Sciences, 11(19), p.8993.
Mahoney, T., Varshney, K. and Hind, M., 2020. AI fairness. O'Reilly Media, Incorporated.
McKinney, W., 2022. Python for data analysis: Data wrangling with pandas, numpy, and jupyter. " O'Reilly Media, Inc.".
Mohammadi, H., Bagheri, A., Giachanou, A. and Oberski, D.L., 2025. Explainability in Practice: A Survey of Explainable NLP Across Various Domains. arXiv preprint arXiv:2502.00837.
Omar, H.K., Frikha, M. and Jumaa, A.K., 2024. PyTorch and TensorFlow Performance Evaluation in Big data Recommendation System. Ingenierie des Systemes d'Information, 29(4), p.1357.
Paparistodimou, G., Duffy, A., Whitfield, R.I., Knight, P. and Robb, M., 2020. A network science-based assessment methodology for robust modular system architectures during early conceptual design. Journal of Engineering Design, 31(4), pp.179-218.
Porter Jr, J., 2020. Improving Quality of Job Application Pre-Processing with Knowledge Graphs. Pace University.
Singh, P., 2024. Media 2.0: A Journey through AI-Enhanced Communication and Content. Media and Al: Navigating, 127.
Siswanto, E., 2024. Leveraging Machine Learning for Personalized Recommendations in Mobile Tourism: A Study on Collaborative and Content-Based Filtering. Journal of Technology Informatics and Engineering, 3(2), pp.235-248. .
Sivasubramaniam, K., 2024. Applying Generative AI in the Job Finding Process (Master's thesis, NTNU).
Wang, X.H., Wang, H.P. and Lai, W.Y., 2022. Sustainable career development for college students: An inquiry into SCCT-based career decision-making. Sustainability, 15(1), p.426.
Widayanti, R., Chakim, M.H.R., Lukita, C., Rahardja, U. and Lutfiani, N., 2023. Improving recommender systems using hybrid techniques of collaborative filtering and content-based filtering. Journal of Applied Data Sciences, 4(3), pp.289-302