University of Mohamed Khider – BISKRA

Faculty of Exact Sciences, Science of Nature and Life

Computer Science Department

Thesis

Presented to obtain the academic master's de gree in

Computer science

AI-powered Job Recommendation Platform and Mobile Application

By *DJEFAFLA Hachem* and *BOUDIBI Abdelhaq*

Directed by *Dr SAHRAOUI Somia*

2023/2024

Members of the jury:

Dr. Terrissa Sadek Labib,	*Président*
Dr. Sahraoui Soumia,	*Encadrant*
Dr. Sahli Sihem,	*Examinateur*

Abstract

Recently, Artificial Intelligence (AI) has transformed various industries, significantly impacting the recruitment sector. This thesis details the development of an AI-powered Job Recommendation Platform and Mobile Application specifically designed for the Algerian job market. By leveraging advanced machine learning algorithms and natural language processing (NLP) techniques, the platform offers a highly efficient and personalized job search experience. It analyzes extensive datasets, including user profiles, job descriptions, and interaction patterns, to provide tailored job recommendations that closely match individual user preferences and qualifications.

In order to recommend relevant jobs, this research presents a content-based recommendation approach using TF-IDF, Bag of Words, and word embeddings. A series of experiments are designed, starting with TF-IDF vectors and cosine similarity as the baseline. Further experiments utilize Bag of Words and word embeddings to observe and compare the results. The objective is to assess how well these different text representation techniques perform in providing accurate and relevant job recommendations, and to determine how each method impacts the quality of job matching. The datasets used for these experiments are provided by Kaggle, ensuring a robust evaluation of the proposed recommendation system.

Keywords: Recruitment, Job recommendation, Artificial intelligence, Natural language processing, Content-based filtering, Text to feature, Cosine similarity.

ملخص

مؤخراً، حوّل الذكاء الاصطناعي AI العديد من الصناعات، مؤثراً بشكل كبير على قطاع التوظيف. تتناول هذه الأطروحة تطوير منصة توصية وظائف مدعومة بالذكاء الاصطناعي وتطبيق محمول مصمم خصيصا لسوق العمل الجزائري. من خلال الاستفادة من خوارزميات التعلم الآلي المتقدمة وتقنيات معالجة اللغة الطبيعية NLP، توفر المنصة تجربة بحث عن وظائف فعالة وشخصية للغاية. تقوم بتحليل مجموعات بيانات واسعة تشمل ملفات المستخدمين، أوصاف الوظائف، وأنماط التفاعل لتقديم توصيات وظائف مخصصة تتوافق بشكل وثيق مع تفضيلات ومؤهلات الأفراد.

لتوصية الوظائف ذات الصلة، يقدم هذا البحث نهجا للتوصية المعتمدة على المحتوى باستخدام TF-IDF ، Bag of Words، و word embeddings. تم تصميم سلسلة من التجارب، بدءاً من متجهات TF-IDF وتشابه جيب التمام كخط أساس. تستخدم التجارب اللاحقة Bag of Words و word embeddings لمراقبة النتائج ومقارنتها. الهدف هو تقييم مدى أداء تقنيات تمثيل النص المختلفة هذه في تقديم توصيات وظائف دقيقة وذات صلة، وتحديد كيفية تأثير كل طريقة على جودة المطابقة الوظيفية. يتم توفير مجموعات البيانات المستخدمة في هذه التجارب من قبل Kaggle، مما يضمن تقييما قويا لنظام التوصية المقترح.

الكلمات المفتاحية:التوظيف، توصيات الوظائف، الذكاء الاصطناعي، معالجة اللغة الطبيعية، التصفية المعتمدة على المحتوى، تحويل النص إلى ميزات، تشابه جيب التمام.

Acknowledgements

First and foremost, I thank my Allah for His grace which has enabled us to successfully conduct this thesis.

I would also like to express my sincere gratitude to my parents. Their unwavering love, support, and encouragement have been my constant source of motivation. I am deeply indebted to them for their sacrifices and for always believing in me.

My deepest appreciation goes to my esteemed advisor, Professor Sahraoui Somia. Thank you for your invaluable guidance, knowledge, and patience, which have been instrumental in shaping this thesis. Your challenges to think critically and the support provided have been essential in navigating this research project.

Lastly, I extend my thanks to my friends and colleagues. Their collaborative spirit and the dynamic, enjoyable atmosphere they fostered were crucial in making my time both productive and pleasurable.

DJEFAFLA Hachem

First and foremost, all praise and thanks are due to Allah for granting me the strength, knowledge, and perseverance to complete this thesis.

I am profoundly thankful to my parents for their unwavering support and encouragement throughout my academic journey. Their belief in my potential has been a constant source of strength and motivation.

Special thanks are due to my advisor Sahraoui Somia, whose expertise and guidance have been invaluable. She has been a mentor and a role model, providing insightful feedback and encouragement at every stage of this project. I am also grateful to the members of my thesis committee.

I must also acknowledge my friends and colleagues, who have provided both technical support and moral encouragement. Their companionship during long hours of work has made this journey enjoyable and fulfilling.

This project was not only an academic challenge but also a personal growth journey, and I am thankful to everyone who played a part in it.

Boudhibi Abdelhaq

Contents

List of Figures

Listings

List of Tables

CHAPTER 1

General introduction

In an era marked by rapid technological advances, the job search process is undergoing a transformative shift through the integration of artificial intelligence (AI) [1]. Our project, an AI-powered Job Recommendation Platform and Mobile App, is at the forefront of this innovation, specifically tailored to address the unique challenges and opportunities within the Algerian job market. This platform leverages sophisticated AI algorithms to not only streamline the job matching process but also enhance the recruitment landscape by ensuring that both job seekers and employers can find their perfect matches with unprecedented efficiency.

Our vision is to revolutionize job searching in Algeria by creating a seamless, intuitive, and highly effective platform that empowers Algerian professionals and companies. By harnessing the power of AI, the platform analyzes vast amounts of data to provide personalized job recommendations, reduces the time and effort typically associated with job hunting, and opens up new pathways for career advancement. This is achieved through a user-centric approach that prioritizes personalized experiences, high engagement, and satisfaction levels among its users.

The mission of this initiative is to foster a dynamic and equitable job market where every individual has the opportunity to achieve their career potential and contribute to economic growth. Through this platform, we aim to provide valuable insights into the labor market, facilitate effective talent acquisition, and support the ongoing development of Algeria's human capital. By addressing specific user needs with customized tools and features, the platform not only caters to individual job seekers and employers but also builds a robust infrastructure that supports broader economic and social goals.

1 Problem statement

The job search and recruitment process in Algeria, as in many parts of the world, faces significant challenges that hinder both job seekers and employers. These challenges form the core problem that our AI-powered Job Recommendation Platform seeks to address:

1. **Inefficiency in Matching:** Traditional job search methods often result in inefficient matching between job seekers and job opportunities. This mismatch is due to a lack of precise tools to analyze and understand the skills, experiences, and preferences of job seekers in relation to available positions.

2. **Limited Access:** Many individuals in Algeria do not have comprehensive access to job opportunities, especially in remote or underserved areas. This limitation restricts their ability to find suitable employment that matches their qualifications and career aspirations.

3. **Time-Consuming Processes:** Both job seekers and employers experience prolonged and often cumbersome recruitment processes. Job seekers spend considerable time applying for jobs without receiving feedback, while employers sift through large volumes of applications, which may not always align with the job criteria.

4. **Economic Inefficiencies:** The unemployment rate, particularly among youth and recent graduates, remains high. This situation is exacerbated by a job market that does not efficiently leverage the available human capital, leading to economic inefficiencies and underemployment.

5. **Lack of Personalization:** Current platforms often do not provide personalized job recommendations based on detailed individual profiles. This lack of personalization results in a generic approach to job recommendations, which fails to meet the specific needs of job seekers or the detailed criteria of employers.

Our platform aims to revolutionize this landscape by implementing advanced AI algorithms that can learn from user interactions, preferences, and profiles to make highly accurate and personalized job recommendations. By doing so, it seeks to significantly reduce the time and effort involved in the job search and hiring processes, improve access to job opportunities, and ultimately enhance the economic vitality of the Algerian job market.

2 Objectives and goals

The primary objective of our AI-powered Job Recommendation Platform is to transform the job search and recruitment landscape in Algeria by leveraging artificial intelligence to match job seekers with ideal job opportunities efficiently and effectively. Specific goals for the project include:

1. **Enhance Job Matching Accuracy:** Utilize AI technologies to analyze vast amounts of data from job seekers and employers, ensuring that recommendations are highly accurate and tailored to the needs and qualifications of users.

2. **Reduce Job Search Time:** Significantly decrease the time job seekers spend searching for suitable jobs by providing targeted, relevant recommendations based on their skills, experience, and preferences.

3. **Increase Employment Rates:** By more efficiently matching job seekers with appropriate job opportunities, aim to reduce unemployment rates, particularly among youth and recent graduates in Algeria.

4. **Improve User Experience:** Develop a user-friendly interface that simplifies the job search and application process, making it accessible and convenient for all users, regardless of their location or technical expertise.

5. **Expand Access to Job Opportunities:** Bridge the gap between job seekers and employers, particularly in underserved areas, by offering a platform that reaches a wider audience and provides equal opportunities for all.

6. **Support Economic Growth:** By optimizing the recruitment process and enhancing job market efficiency, contribute to the overall economic development of Algeria, fostering a dynamic, skilled workforce that can drive innovation and productivity.

7. **Foster Continuous Learning and Improvement:** Continuously update and refine the AI algorithms based on user feedback and interaction data to improve the effectiveness of the recommendations and adapt to changing job market conditions.

These goals are designed to address the specific challenges identified in the problem statement, aligning with our mission to empower individuals and companies in Algeria through innovative, AI-driven job recommendations.

3 Dissertation organization

The organization of the dissertation is as follows:

1. **Chapter 1:** General Introduction: This introductory chapter outlines the problem statement, objectives, and goals of the project. It provides an overview of the dissertation's structure, setting the stage for the subsequent chapters.

2. **Chapter 2:** Background: This chapter establishes the theoretical and contextual framework for job recommendation systems. It explores the role of technology in modern recruitment, defines recommendation systems, and discusses their types, benefits, and applications. Additionally, it delves into the use of Natural Language Processing (NLP) for recommender systems, covering various NLP techniques and their relevance to the project.

3. **Chapter 3:** Design and Implementation of Recommendation System: This chapter focuses on the technical development of the recommendation system. It details the system design, including dataset description, data preparation, preprocessing, and evaluation metrics. The implementation section covers the frameworks, tools, and libraries used, along with the process of data loading, preparation, and feature extraction.

4. **Chapter 4:** Experiments and Results: This chapter assesses the effectiveness of the AI algorithms in making accurate job recommendations, discusses the experiment environment, and evaluates the results and their implications.

5. **Chapter 5:** Platform Development and Recommendation System Integration: This chapter details the system architecture, including the architecture of the platform, components and their interactions, and the technology stack used. It covers both frontend and backend development, discussing the clean architecture, MVC architecture, and API development. Additionally, it explores security considerations and additional features such as chat-bots, voice & video calling, and real-time messaging.

6. **Chapter 6:** Conclusion: In this last chapter, we summarize our approach to developing an AI-powered job recommendation platform.

Chapter 2
Background

1 Introduction

This chapter provides the contextual foundation for understanding the AI-powered Job Recommendation System. It delves into the technological and market landscapes that necessitate such a platform, particularly within the Algerian context. The chapter begins by exploring the role of technology in modern recruitment, highlighting the challenges faced by traditional recruitment methods and how recommendation systems have evolved to address these issues.

The chapter also introduces the concept of recommendation systems, discussing their origins, different types, and how they function to connect job seekers and employers. Furthermore, it examines the integration of Natural Language Processing (NLP) and Machine Learning (ML) techniques into these systems, showcasing how they can enhance job recommendations by analyzing textual data from resumes and job descriptions.

Overall, this chapter sets the stage for the subsequent discussions on system design, implementation, and the impact of the AI-powered platform on the Algerian job market.

2 Role of Technology in Modern Recruitment

In recent years, technological advancements have significantly transformed the recruitment landscape, making the process more efficient, effective, and inclusive [2]. Traditional recruitment methods, often characterized by manual processes, extensive paperwork, and limited reach, are being supplanted by innovative technological solutions that leverage artificial intelligence (AI), machine learning (ML), and data analytics.

- **Enhanced Efficiency and Accuracy** One of the primary roles of technology in modern recruitment is enhancing the efficiency and accuracy of the hiring process. Traditional recruitment often involves sifting through large volumes of resumes manually, a time-consuming and error-prone task. AI-powered applicant tracking systems can automate this process by scanning resumes for relevant keywords and qualifications, significantly reducing the time required to shortlist candidates. These systems use natural language processing (NLP) to understand the context of resumes and job descriptions, ensuring that only the most suitable candidates are selected for further evaluation [2] .

- **Personalized Candidate Matching** Technology has also enabled personalized candidate matching, a significant improvement over the generic approaches of the past. Recommendation systems, similar to those used by e-commerce platforms, analyze a candidate's skills, experience, and preferences to suggest the most relevant job opportunities. This personalized approach ensures that candidates are matched with roles that align closely with their qualifications and career aspirations, improving job satisfaction and retention rates [3].

- **Overcoming Geographic and Demographic Barriers** Modern recruitment technologies have made it possible to reach a wider and more diverse pool of candidates. Online job portals and social media platforms allow companies to advertise job openings to a global audience, overcoming geographic limitations. Additionally, technologies like video interviewing platforms enable recruiters to conduct remote interviews, making it easier to evaluate candidates from different locations. This inclusivity ensures that companies can access a broader talent pool, enhancing diversity within the workforce [2].

- **Data-Driven Decision Making** Data analytics plays a crucial role in modern recruitment by providing insights into various aspects of the hiring process. Companies can use analytics to track the effectiveness of their recruitment strategies, identify bottlenecks in the hiring process, and understand the characteristics of successful hires. This data-driven approach allows for continuous improvement and optimization of recruitment practices, leading to better hiring outcomes.

- **Improved Candidate Experience** Technological advancements have also improved the overall candidate experience. AI-powered chatbots, for example, can provide instant responses to candidate inquiries, guide them through the application process, and keep them updated on the status of their application. These technologies ensure that candidates feel valued and engaged throughout the recruitment process, enhancing the employer brand and attracting top talent [3].

While technology has brought numerous benefits to recruitment, it also presents certain challenges. The use of AI and ML in recruitment must be managed carefully to avoid biases that can arise from the training data. Ensuring transparency and fairness in AI-driven recruitment processes is essential to maintaining trust and integrity. Companies must also be mindful of data privacy regulations and ensure that candidate information is handled securely and ethically.

3 What is a Recommendation System

3.1 Definition

"A recommendation system (or recommender system) is a class of machine learning that uses data to help predict, narrow down, and find what people are looking for among an exponentially growing number of options" [4].

A recommendation system, also known as a recommender system, is a subclass of information filtering systems that aims to predict and identify items that a user may be interested in. It is a form of machine learning that analyzes data to provide recommendations tailored to the user's preferences and behavior patterns. The data can range from structured entries to complex user behaviors, enabling these systems to navigate through an expanding array of choices to find personalized options for users [5].

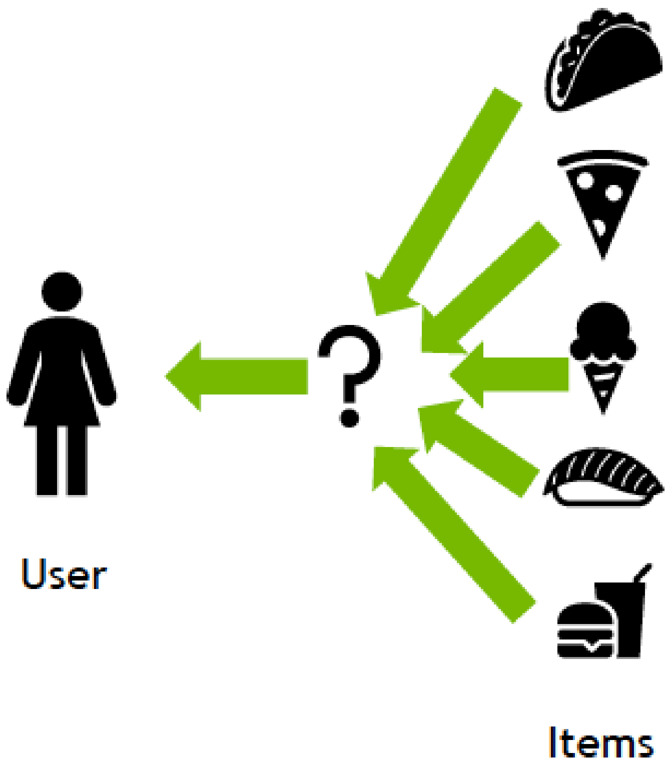


Fig. 2.1: Visual Representation of a Recommendation System.

3.2 The Beginning of Recommendation System

Recommendation systems have their roots in the late 20th century, with the emergence of the internet as a catalyst for digital content proliferation. The foundational concept of collaborative filtering was introduced with the Tapestry system in 1992, designed to filter emails and news based on user feedback. The term "collaborative filtering" was coined, leading to the creation of GroupLens in 1994, a system that recommended news articles on Usenet by analyzing readers' ratings [6].

One of the landmark events in the evolution of recommendation systems was the development of Amazon's recommendation algorithm in the late 1990s, which applied item-to-item collaborative filtering to personalize user experiences. This was closely followed by the Netflix Prize competition in 2006, which incentivized the improvement of algorithm accuracy for movie recommendations, spotlighting the potential of advanced machine learning in this field [7].

Through the early 2000s, the refinement of matrix factorization techniques and the introduction of singular value decomposition (SVD) enabled more precise user-item interaction predictions. By the 2010s, the integration of deep learning models marked the next leap forward, with systems like YouTube's recommendation engine adopting complex neural networks to match users with content [6].

Today's sophisticated recommendation systems trace their lineage back to these pivotal developments, now leveraging vast arrays of user data and advanced predictive analytics to curate personalized experiences across the digital landscape.

3.3 Use Cases and Applications

Recommendation systems have a broad spectrum of applications across various sectors, leveraging their capacity to enhance user interaction and satisfaction by personalizing content and suggestions. Here are some prominent use cases:

1. **E-commerce and Retail:** Online retailers like Amazon use recommendation systems to suggest products to customers based on browsing and purchase history, increasing both customer satisfaction and sales [8].

2. **Media and Entertainment:** Platforms such as Netflix and Spotify recommend movies, TV shows, and music based on users' past consumption patterns. This not only improves user experience but also boosts engagement and retention rates (Underwood) [9].

3. **Social Media:** Social networking sites like Facebook and LinkedIn utilize recommendation systems to suggest friends, jobs, and content to users, thereby enhancing connectivity and relevance of the network experience [9].

4. **Finance and Banking:** Recommendation systems help financial services provide personalized financial advice and product recommendations like credit cards, loans, and investment options based on customer behavior and preferences [9].

5. **Healthcare:** In healthcare, recommendation systems can suggest personalized treatment plans and medications by analyzing patient history and similar cases, thereby improving outcomes through tailored care strategies [9].

6. **Job Recommendation:** Platforms like LinkedIn and Glassdoor use recommendation systems to match job seekers with job postings that align with their skill sets and career interests, streamlining the recruitment process [9].

7. **Tourism and Hospitality:** Recommendation systems in these sectors suggest travel destinations, hotels, and recreational activities based on user preferences and previous trips, enhancing the personalized travel planning experience [9].

These applications showcase the versatility and impact of recommendation systems, illustrating their pivotal role in transforming user interactions across digital platforms by providing targeted, relevant, and timely suggestions.

3.4 Benefits of Recommendation Systems

Recommendation systems offer a multitude of benefits across various sectors by personalizing user experiences and improving operational efficiencies. Here are some key advantages:

1. **Increased Sales and Customer Satisfaction:** By suggesting products and services aligned with user preferences and past behavior, recommendation systems enhance customer satisfaction, which often leads to increased sales and customer retention [8].

2. **Improved Content Discovery:** These systems enable users to discover products, services, or content that they might not have otherwise found, enhancing user engagement and satisfaction [10] .

3. **Efficient Information Filtering:** Recommendation systems filter massive amounts of content to present users with items likely to be of interest, saving time and reducing information overload [10] .

4. **Enhanced User Experience:** They personalize the user experience on digital platforms by understanding user preferences and presenting tailored options, thus improving overall user satisfaction and platform usability [10].

5. **Optimized Inventory Management:** In e-commerce, recommendation systems help predict product demand more accurately, assisting in better inventory management and reducing operational costs [10] .

6. **Data-Driven Insights and Decision Making:** By analyzing user interactions and feedback, recommendation systems provide valuable insights that can inform business strategies and marketing campaigns, fostering data-driven decision-making [10].

These systems have transformed how businesses engage with customers, making them essential tools in today's digital economy.

4 Types of Recommendation Systems

Recommendation systems can be classified into several types. Each type utilizes distinct methods to analyze user preferences and item characteristics, catering to specific needs and scenarios in various domains. This section explores the primary types of recommendation systems, namely Content-Based Filtering, Collaborative Filtering, Hybrid Systems, and those based on Deep Learning. We'll discuss how these systems function, their unique advantages, and where they are best applied, illustrating the diverse approaches to recommendation that have evolved to tackle different recommendation challenges. This examination will not only delineate the operational mechanisms but also highlight the contextual applications of each system type, providing a comprehensive understanding of their impact on personalization and recommendation quality in digital platforms.

4.1 Content-Based Filtering

Content-Based Filtering is a type of recommendation system that utilizes the features of items themselves to recommend other similar items to users. This approach relies on the specific characteristics of an item—such as genre, keywords, or attributes—to make recommendations. Content-based filtering operates under the assumption that if a user likes a certain item, they are likely to enjoy other items that are similar in content [5].

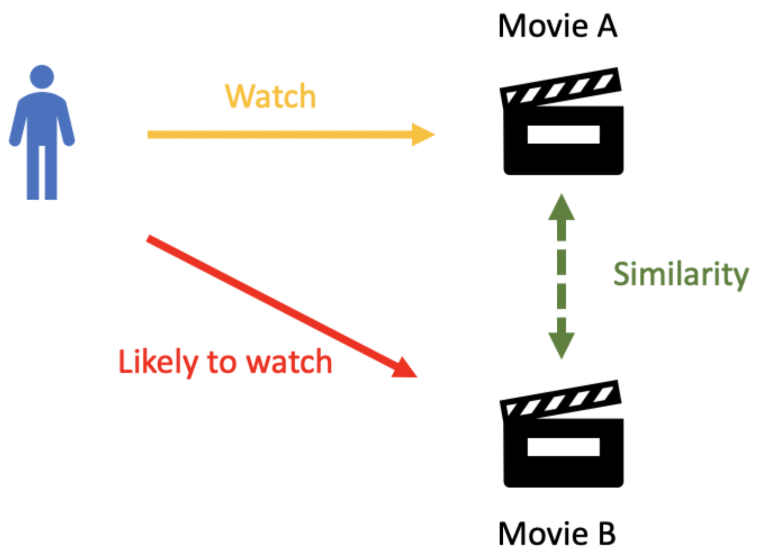


Fig. 2.2: Content-Based Filtering Process.

The system works by creating profiles for both users and items based on item descriptions and user interactions. For instance, in a movie recommendation engine, the attributes can include the director, actors, film genre, and plot keywords. The user profile might incorporate the ratings or preferences expressed for certain films, which the system then uses to calculate similarity scores between the user's profile and the attributes of different movies.

One of the primary advantages of content-based filtering is its ability to recommend items that are lesser-known or new, as it does not require user interaction data for those items. However, a significant drawback is that it can lead to a lack of diversity in the recommendations, as the system tends to suggest items that are closely aligned with the user's existing preferences [11].

Content-based systems are particularly useful when detailed metadata about items is available, allowing for nuanced differentiation and precise matching based on content similarity [11].

4.2 Collaborative Filtering

Collaborative Filtering (CF) is one of the most popular and widely implemented types of recommendation systems. Unlike content-based filtering, which focuses on the attributes of items, collaborative filtering builds recommendations based on the relationships and interactions between users and items. The fundamental premise of CF is that if users agreed in the past, they are likely to agree again in the future [5].

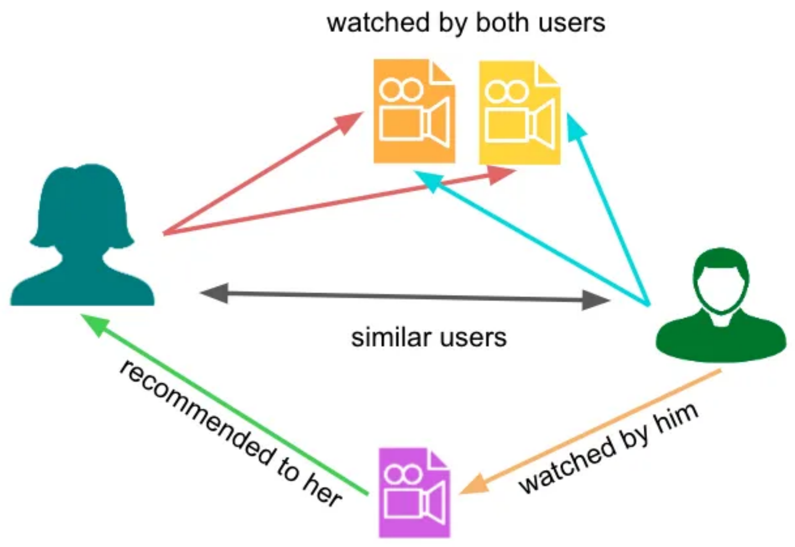


Fig. 2.3: Collaborative Filtering Process.

There are two main approaches within collaborative filtering:

1. **User-based Collaborative Filtering:** This method involves finding users who have similar preferences and tastes as a given user and recommending items that these similar users have liked. For instance, if User A has similar movie preferences to User B, and User B likes a particular movie that User A hasn't seen yet, that movie will be recommended to User A [5].

2. **Item-based Collaborative Filtering:** This approach focuses on the similarity between items rather than users. It recommends items that are similar to those a user has already shown an interest in. For example, if a user has rated a movie highly, the system will recommend movies that other users who liked the same movie also rated highly [5]).

Collaborative filtering systems are particularly effective because they require no item metadata and purely operate on the interactions recorded between users and items, which can lead to discovering unexpected preferences and serendipitous recommendations. However, they face challenges such as cold start for new users or items with no previous interactions, and scalability due to the need to compute relationships between all users or items [11].

This approach's popularity stems from its ability to leverage the collective tastes of the user community, thereby providing personalized and dynamically updated recommendations based on user interactions [11].

4.2.1 Matrix Factorization for Recommendation

Matrix Factorization is a key technique in collaborative filtering systems, particularly useful for extracting latent factors from user-item interactions. This approach helps to address the issue of scalability and sparsity in large datasets, making it an essential tool for recommendation engines.

The technique works by decomposing a large, sparse user-item matrix (where rows represent users, columns represent items, and values represent interactions such as ratings or views) into two lower-dimensional matrices: a user matrix and an item matrix. These matrices capture latent factors that can be used to reconstruct the original interactions.

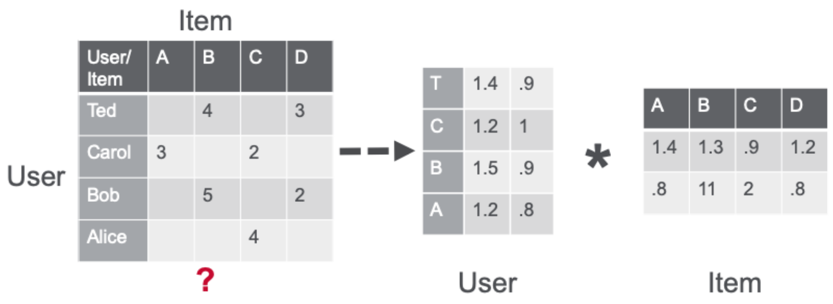


Fig. 2.4: Matrix Factorization Process in Recommendation System.

1. **Latent Factors:** These represent abstract concepts or characteristics that underlie user preferences and item attributes. For example, in the context of movie recommendations, latent factors might capture dimensions such as genre preference, acting quality, or directorial style.

2. **Singular Value Decomposition (SVD):** A common form of matrix factorization, SVD breaks down the original matrix into three matrices, effectively reducing its dimensionality. This helps identify patterns in user preferences and item characteristics.

3. **Alternating Least Squares (ALS):** Another matrix factorization technique that iteratively minimizes the difference between predicted and actual interactions, refining the user and item matrices to improve recommendation accuracy.

By uncovering these latent factors, matrix factorization allows collaborative filtering systems to make more accurate predictions and recommendations, enhancing the user experience. This technique also mitigates the cold start problem to some extent by filling in missing values in the user-item matrix through approximations based on similar users or items.

4.3 Hybrid Recommendation Systems

Hybrid Recommendation Systems combine the strengths of multiple recommendation techniques, typically merging collaborative filtering, content-based filtering, and other methods, to enhance recommendation accuracy and overcome the limitations inherent in any single approach. By integrating different systems, hybrid models can provide more robust, relevant, and diversified recommendations [11] .

There are several ways to design hybrid systems:

1. **Weighted:** This method combines the predictions of various recommendation techniques, weighting each according to its reliability or accuracy in specific contexts.

2. **Switching:** The system switches between recommendation strategies based on the situation. For instance, it might use content-based filtering when sufficient user preference data is unavailable and switch to collaborative filtering once enough data is collected.

3. **Mixed:** Recommendations from different approaches are presented together. For example, the top recommendations from both collaborative and content-based filters could be displayed simultaneously to the user.

4. **Feature Combination:** This method involves integrating features derived from different recommendation techniques into a single model. For example, using both user-item interaction history and item metadata as inputs to a machine learning model.

5. **Model Ensemble:** Similar to feature combination, model ensembles use machine learning algorithms to combine multiple recommendation models into a single predictive model, aiming to capitalize on the unique strengths of each underlying model.

Hybrid systems are particularly effective because they can personalize the user experience by addressing a wider range of user behaviors and preferences. They also help alleviate the cold start problem by using content-based methods when user ratings are scarce and collaborative methods when metadata is insufficient.

By leveraging the complementary advantages of various recommendation models, hybrid systems can significantly enhance the performance and flexibility of recommender engines, making them adaptable to complex and evolving user needs.

4.4 Deep Learning Based Recommender Systems

Deep Learning Based Recommender Systems utilize advanced neural network architectures to model complex user-item interactions and uncover deep patterns in large datasets. These systems represent the cutting-edge in recommendation technology, leveraging the power of deep learning to provide highly personalized recommendations [5].

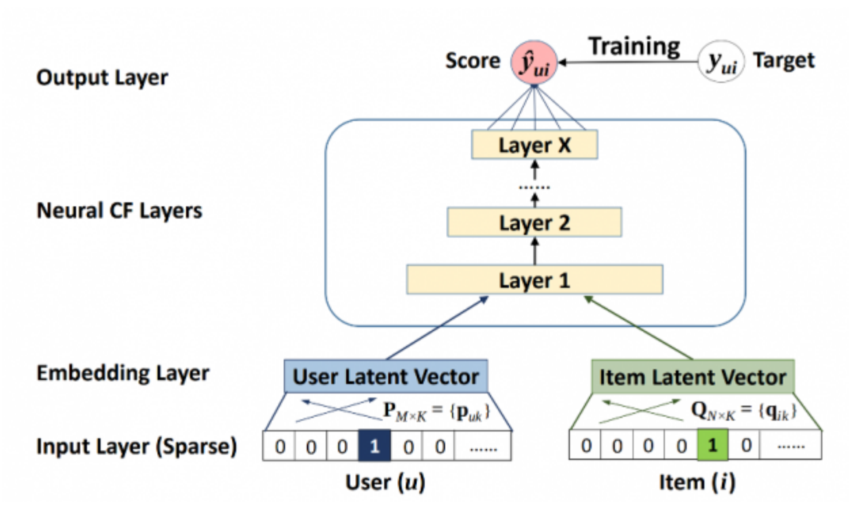


Fig. 2.5: Architecture of a Deep Learning-Based Recommendation System.

The core advantage of deep learning models lies in their ability to learn feature representations automatically from raw data, which significantly enhances the capability of recommendation systems in several ways:

1. **Feature Learning:** Deep learning models can learn intricate structures in the data autonomously, without requiring manual feature engineering. This capability allows them to capture subtle user preferences and item attributes that traditional methods might overlook.

2. **Complex Interactions:** They are capable of modeling non-linear and complex interactions between users and items, which are often challenging to capture with traditional matrix factorization techniques.

3. **Scalability and Flexibility:** Modern deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can scale effectively to accommodate the vast amount of data typically found in commercial recommendation systems. They are also adaptable to various types of data input, including images, text, and sequential interaction data.

4. **Enhanced Contextual Relevance:** By integrating contextual information such as time, location, and social media behavior, deep learning models can offer dynamic and context-aware recommendations.

Popular models and approaches include:

- **Neural Collaborative Filtering (NCF):** This model combines traditional collaborative filtering with neural networks to enhance the interaction modeling capabilities.
- **Autoencoders:** Used for unsupervised learning of efficient codings, helping in noise reduction and dimensionality reduction for better recommendation quality.
- **Sequence Models like LSTM (Long Short-Term Memory):** Effective for sequential recommendation scenarios such as next-item recommendation where the order of interactions matters.

Deep Learning Based Recommender Systems are increasingly prevalent in sectors where the accuracy of recommendation is critical and data is abundant, such as streaming services, e-commerce, and social networks. Their ability to improve over time with more data makes them especially valuable in rapidly evolving markets.

5 NLP for Recommender Systems

5.1 What is NLP

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) concerned with enabling computers to comprehend and manipulate human language. NLP leverages machine learning techniques to endow computers with the ability to interpret, manipulate, and even generate human language. This field plays a crucial role in bridging the communication gap between humans and machines [12].

NLP has opened doors to a world where machines can understand and interact with human language. This capability translates into a wide range of applications that are transforming various industries [13] . Here are some specific areas where NLP finds applications:

1. **Language Translation:** Breaking down language barriers by translating text between languages. (Applications: communication tools, travel assistance) [13].
2. **Virtual Assistants:** Understanding Your Needs Through Conversation. (Applications: Siri, Alexa, Google Assistant) - NLP empowers virtual assistants to understand spoken language (speech recognition), grasp the intent behind your requests (natural language understanding), and respond in a natural way (natural language generation) [12].
3. **Search Engines:** Going Beyond Keywords to Find What You Really Need. (Applications: Google Search, Bing) - NLP allows search engines to understand the meaning and context of your queries (information retrieval), enabling them to return the most relevant results even if the exact keywords aren't present [13].
4. **Sentiment Analysis:** Unveiling Emotions Behind the Text. (Applications: Social media monitoring, customer review analysis) - NLP empowers tools to analyze text and determine the emotional tone (positive, negative, neutral). This helps businesses understand customer sentiment and brand perception [13].

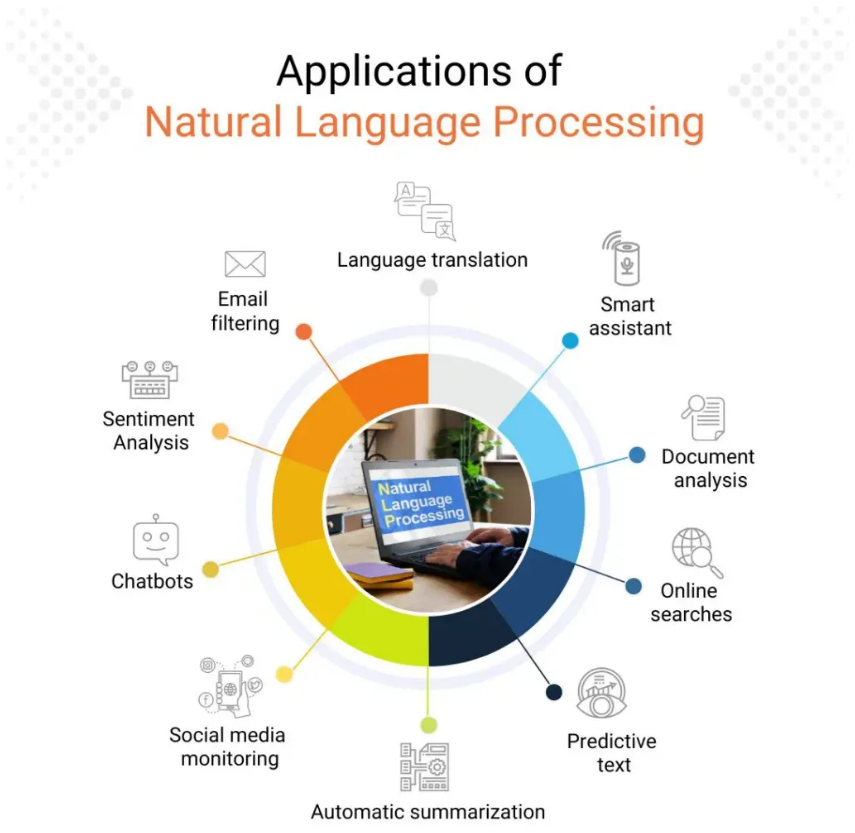


Fig. 2.6: Diverse Applications of Natural Language Processing.

5. **Spell and Grammar Checking:** Ensuring Clear and Error-Free Communication. (Applications: Word processors, email clients) - NLP techniques break down text into words (tokenization) and assign grammatical labels (part-of-speech tagging) to analyze structure and spelling. By comparing the text to a large dictionary and grammar rules, NLP helps identify and suggest corrections for errors [13].

6. **Text Summarization:** Condensing lengthy text documents into concise summaries. (Applications: news aggregators, research assistants) [14].

7. **Information Retrieval (IR):** While traditionally keyword-based, NLP is increasingly being used in information retrieval systems to improve search accuracy by understanding the meaning and context of user queries [12].

8. **Information Extraction (IE):** Extracting Meaningful Information from Text. (Applications: question answering, data mining) [14].

9. **Chatbots and Virtual Assistants:** Providing interactive customer service or information retrieval experiences through natural language conversations. (Applications: e-commerce websites, banking apps) [13].

In NLP tasks, raw textual data is often messy and unsuitable for direct analysis by machines. To address this, the upcoming sections will delve into two crucial preliminary steps: Text Preprocessing and Text Representation.

5.2 Text Preprocessing

Text preprocessing is a crucial first step in any NLP task. It involves transforming raw text data into a clean and structured format that machines can understand and process effectively. This process typically involves several sub-steps [15], each addressing a specific aspect of the raw text:

5.2.1 Tokenization and Text Cleaning

1. **Tokenization:** This is the process of splitting text data into smaller units of meaning, often referred to as tokens. These tokens can be individual words, phrases, or even sentences depending on the specific NLP task.

2. **Text Cleaning:** This step aims to remove noise and inconsistencies from the text data. It may involve tasks like :

   - **Lowercasing:** Converting all text to lowercase to ensure consistency and avoid treating capitalization as a separate feature.
   - **Punctuation Removal:** Removing punctuation marks like commas, periods, exclamation points, etc., as they often don't hold significant meaning for the NLP task at hand.
   - **Symbol Removal:** Removing symbols like @, \$, %, &, etc., which might not be relevant to the analysis.
   - **Extra Space Removal:** Removing unnecessary whitespace characters like multiple spaces or tabs to improve data consistency.

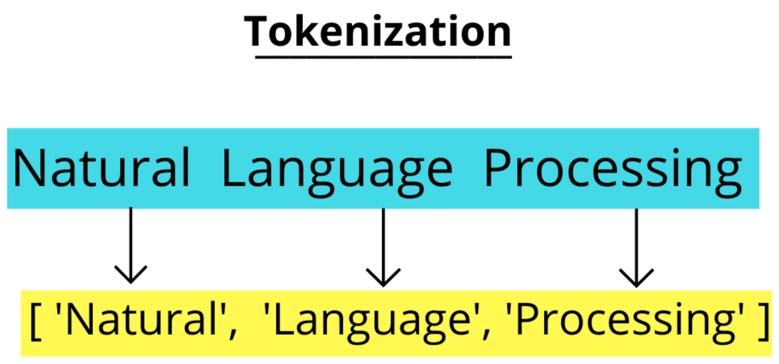

Tokenization
Natural Language Processing
['Natural', 'Language', 'Processing']

Fig. 2.7: Tokenization Process in NLP.

Text cleaning can encompass additional tasks depending on the specific needs of the NLP application.

By performing tokenization and text cleaning, we prepare the text data for further processing and analysis in NLP tasks. This ensures that the model focuses on the core content of the text and avoids getting sidetracked by irrelevant elements.

5.2.2 Stop Words Removing

After breaking text into tokens and cleaning it, stop words are often addressed in text preprocessing.

Stop Words: These are extremely common words like "the," "in," or "and" that carry little meaning on their own within a document. While stop words play a grammatical role, they don't significantly contribute to the core meaning of the text. Removing them offers benefits :

- **Reduced Processing Time:** Eliminating stop words lessens the data processed by NLP models, leading to faster training and analysis.
- **Improved Focus:** By removing stop words, NLP models can focus on the more content-rich words, potentially improving accuracy in tasks like sentiment analysis or topic modeling.

5.2.3 Stemming and Lemmatizing

Following stop word removal, text preprocessing often tackles the task of reducing words to their base forms. Here, we encounter two techniques: stemming and lemmatization [15].

1. **Stemming:** This process aims to simplify words by chopping off prefixes or suffixes to obtain their root form (stem). For example, "running" would be stemmed to "run," and "plays" would be stemmed to "play." Stemming is a rule-based approach that doesn't always result in a valid word (e.g., stemming "agrees" might result in "agree" which isn't the most accurate root form). However, it's computationally efficient.
2. **Lemmatization:** This technique goes a step further than stemming. It considers the grammatical context of a word and aims to convert it to its dictionary or base form (lemma), ensuring a valid word. For example, both "running" and "runs" would be lemmatized to "run." Lemmatization is more linguistically accurate but can be computationally more expensive than stemming.

5.2.4 Part-of-Speech Tagging

Moving beyond word forms, text preprocessing can delve into the grammatical role of each word using part-of-speech (POS) tagging.

Word	Stemming	Lemmatization
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

Tab. 2.1: Comparing Stemming and Lemmatization Results for Different Words

This process assigns a grammatical category (such as noun, verb, adjective, adverb, etc.) to each word in a sentence. This provides a deeper understanding of the sentence structure and how words function within it [15].

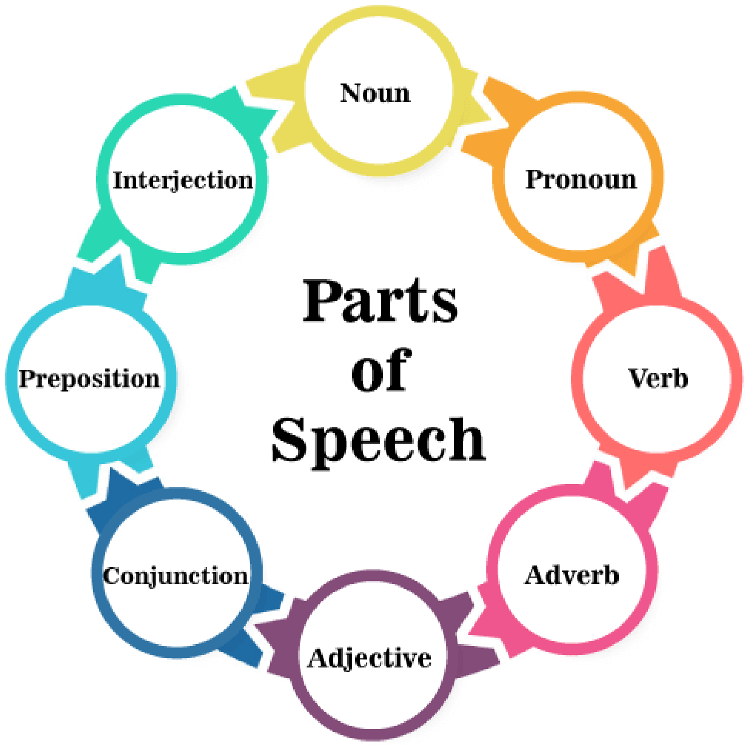


Fig. 2.8: Grammatical categories.

5.2.5 Named Entity Recognition (NER)

This technique focuses on identifying and classifying specific types of entities within a text corpus. These entities can be people, organizations, locations, dates, monetary values, or other predefined categories relevant to the NLP task [15] .

In the sentence "Barack Obama, the former president of the United States, visited Paris last week," NER might identify "Barack Obama" as a Person, "United States" as a Location, and "Paris" as a Location.

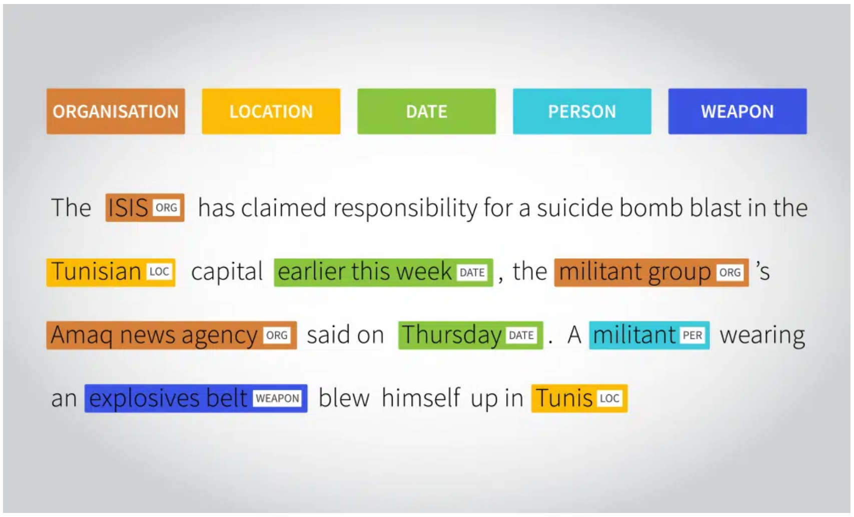

ORGANISATION
LOCATION
DATE
PERSON
WEAPON
The ISIS ORG has claimed responsibility for a suicide bomb blast in the Tunisian LOC capital earlier this week DATE, the militant group ORG 's Amaq news agency ORG said on Thursday DATE. A militant PER wearing an explosives belt WEAPON blew himself up in Tunis LOC

Fig. 2.9: Example of Named Entity Recognition (NER) in NLP.

By incorporating NER into text preprocessing, we can equip NLP models with the ability to recognize and extract crucial information from text data, leading to a deeper understanding of the content.

5.3 Text Representation

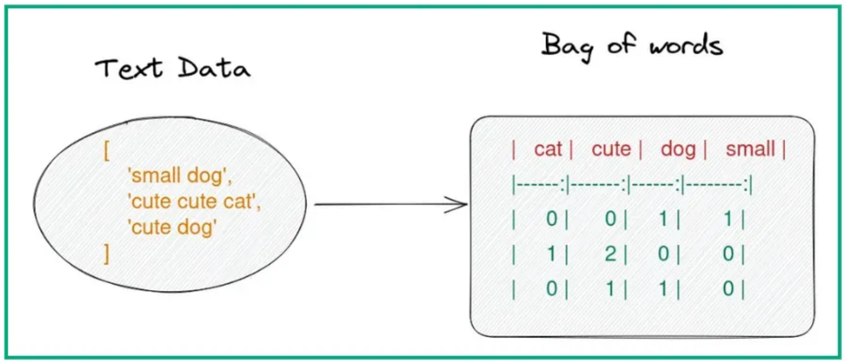
Text representation is a fundamental concept in Natural Language Processing (NLP). It refers to the process of converting raw text data into a numerical format that machines can understand and process effectively. This numerical representation captures the essential aspects of the text, such as word meaning, relationships between words, and overall sentiment [16].

Just like we use indexes and catalogs to navigate vast libraries of information, text representation allows NLP models to efficiently analyze and extract meaning from textual data.

In the following sections, we will explore several established techniques for representing text data numerically:

5.3.1 Bag-of-Words (BoW) Method

The Bag-of-Words (BoW) technique represents text by focusing on word presence or frequency within a document. Think of it like a bag of unique words, where order doesn't matter. This approach allows for efficient document similarity comparisons and is useful for tasks like spam filtering or basic sentiment analysis based on word frequency. However, BoW disregards word order and context, limiting its ability to capture the full meaning of text [16].



```
[
  'small dog',
  'cute cute cat',
  'cute dog'
]
```

cat	cute	dog	small
0	0	1	1
1	2	0	0
0	1	1	0

Fig. 2.10: Transformation of Text Data into a Bag of Words.

5.3.2 TF-IDF (Term Frequency-Inverse Document Frequency)

The Bag-of-Words (BoW) technique is straightforward and effective, yet it presents an issue by treating all words equally. Consequently, it fails to differentiate between highly frequent or rare words. To address this limitation, TF-IDF (Term Frequency-Inverse Document Frequency) is introduced as a solution.

In contrast to the bag of words model, TF-IDF representation considers the significance of each word within a document. TF stands for term frequency, representing how often a word appears in a document, while IDF stands for inverse document frequency, indicating the rarity of the word across all documents [16] .

To understand TF-IDF,it's essential to discuss the two constituent terms individually:

- **Term Frequency (TF)** TF measures how often a specific word appears in relation to the overall length of the document. Calculated by dividing the frequency of the word by the total number of words in the document, TF offers insights into the prominence of the term, aiding in the comprehension of the document's content.

$$TF(t,d) = \frac{number of times "t" appears in d}{total number of words in "d"} \tag{2.1}$$

- **Inverse document frequency (IDF)** IDF assesses the rarity of a word across a corpus of documents. Common words receive lower IDF values, while rare words get higher values. IDF helps prioritize terms based on their uniqueness and potential informativeness within the corpus.

$$IDF(t) = log(\frac{total number of documents}{number of documents that contain "t"}) \tag{2.2}$$

- **TF-IDF Score** The TF-IDF score for a term in a document is derived by multiplying its TF and IDF values. This composite score indicates the term's significance both within the document and across the corpus. Higher TF-IDF scores denote greater importance of the term in context and across the entire collection of documents.

$$TF_IDF(t,d) = TF(t,d)IDF(t) \tag{2.3}$$

5.3.3 Word Embeddings : Word2Vec

Word embeddings are numerical representations of words. These aren't just random numbers assigned to each word. Instead, they capture the semantic relationships and meaning of words by encoding them as vectors in a high-dimensional space [16] .

Imagine a map where similar words are positioned closer together. Words with opposite meanings might be placed far apart. This way, a computer can process words based on their meaning and relationships, rather than just treating them as isolated symbols.

Word2Vec leverages the idea that words appearing in similar contexts likely share similar meanings. It accomplishes this by analyzing surrounding words and learning vector representations for each word. This essentially creates a numerical map where words with close meanings reside in closer proximity [16].

The technique offers two main approaches:

1. **Continuous Bag-of-Words (CBOW):** This method acts like a guessing game. Given a set of surrounding context words, CBOW predicts the word that would likely appear in the center. This helps the model understand how a word relates to its neighbors in a sentence .

2. **Skip-gram:** Here, the focus shifts to a single word. Skip-gram predicts the surrounding context words based on the given word. This approach helps the model learn the meaning of a word based on the company it keeps within a sentence .

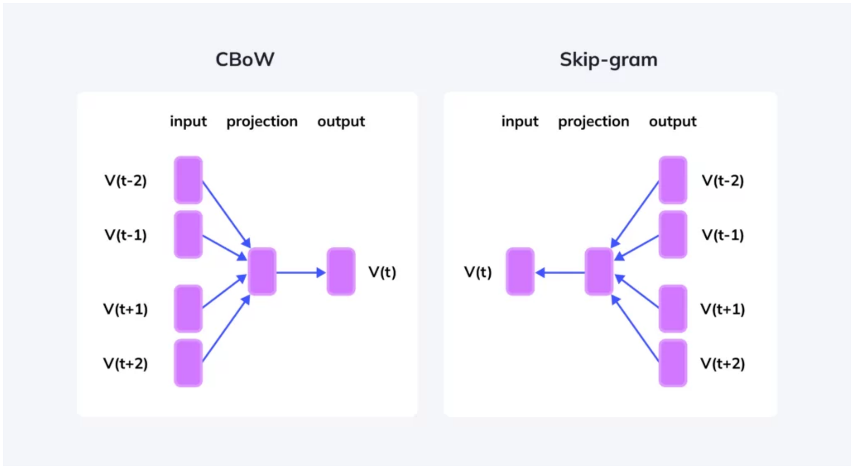


Fig. 2.11: Comparison of CBoW and Skip-gram Models.

5.3.4 Comparative Analysis of 3 Techniques

This section explores the strengths and weaknesses of three text representation techniques: Bag-of-Words (BoW), TF-IDF, and Word2Vec. It highlights their characteristics and identifies the tasks for which each method is best suited.

Technique	**BoW**	**TF-IDF**	**Word2Vec**
Computational Efficiency	Most efficient, ideal for smaller datasets	Moderately demanding	Highly computational (deep learning)
Semantic Representation	Basic, lacks context and word order	Weights word importance, limited context	Excellent, captures semantic meaning and relationships
Suitability for Data Size	Effective with limited data	Effective with limited data	Ideal for larger datasets
Ease of Implementation	Easiest to implement	Moderate linguistic understanding required	Complex setup, deep learning knowledge required
Application Scenarios	Document classification, spam detection, basic sentiment analysis	Information retrieval, keyword extraction, document clustering	Machine translation, contextual search, advanced sentiment analysis

Tab. 2.2: Comparative Analysis of BoW, TF-IDF, and Word2Vec Techniques

5.3.5 NLP with ML

In the realm of Natural Language Processing (NLP), which involves understanding and generating human language, machine learning techniques play a crucial role. Human language is intricate, with numerous nuances and complexities that make it challenging to process effectively. This complexity results in a high-dimensional feature space, which refers to the vast number of potential features or attributes that can be associated with language data [17].

Machine learning models offer an advantage in dealing with this complexity. Unlike rule-based systems, which rely on predefined rules created by experts, machine learning models can learn patterns and relationships directly from data. These models typically have a larger capacity to represent complex relationships within the data.

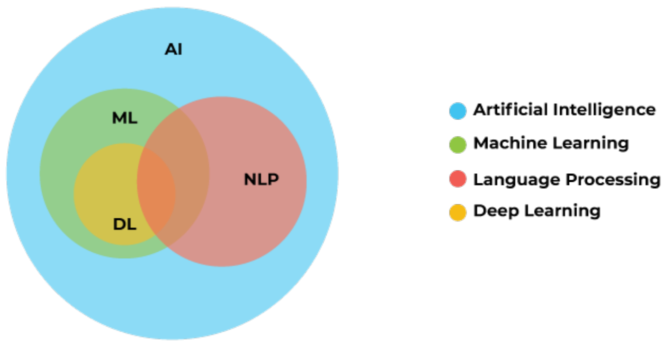


Fig. 2.12: The Intersections of AI, ML, DL, and NLP.

5.4 NLP Approaches for Recommendations

Natural Language Processing (NLP) techniques have emerged as pivotal tools in enhancing recommendation systems within various domains, particularly in job recommendation platforms. These systems leverage NLP to better understand and match job descriptions with candidate profiles by analyzing semantic and syntactic aspects of text data. Key NLP techniques employed encompass text similarity assessment, named entity recognition (NER), topic extraction, keyword identification, and concise text summarization.

By integrating these NLP approaches, recommendation systems can offer more precise, contextually relevant job matches, significantly enhancing user experience and platform efficiency.

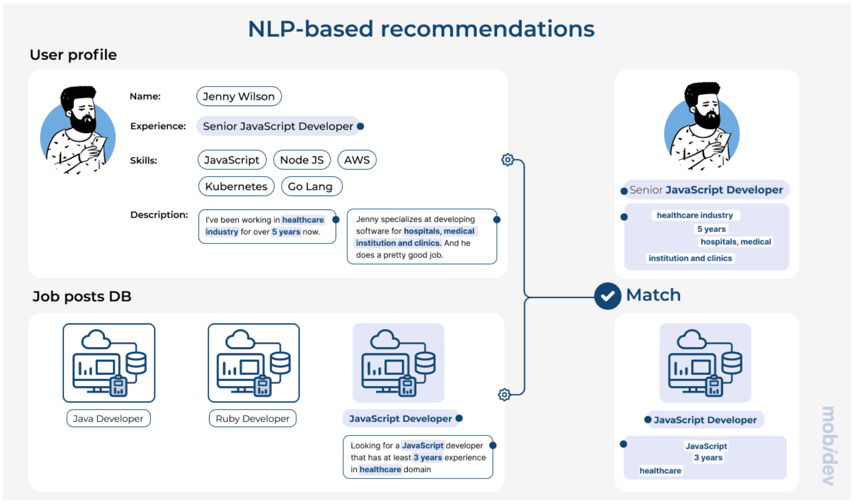

NLP-based recommendations
User profile
Name:
Jenny Wilson
Experience:
Senior JavaScript Developer
Skills:
JavaScript
Node JS
AWS
Kubernetes
Go Lang
Description:
I've been working in healthcare industry for over 5 years now.
Jenny specializes at developing software for hospitals, medical institution and clinics. And he does a pretty good job.
Senior JavaScript Developer
healthcare industry
5 years
hospitals, medical
institution and clinics
Match
Job posts DB
Java Developer
Ruby Developer
JavaScript Developer
Looking for a JavaScript developer that has at least 3 years experience in healthcare domain
JavaScript Developer
JavaScript
3 years
healthcare
mobidev

Fig. 2.13: NLP-Based Recommendations for Job Matching.

5.4.1 Text Similarity

Text similarity is a fundamental Natural Language Processing (NLP) technique pivotal in enhancing recommendation systems, especially in contexts that require matching textual content, such as job recommendations. This technique measures the degree of closeness between two pieces of text, enabling systems to efficiently identify documents most aligned with a user's preferences or queries. Several algorithms for computing text similarity include cosine similarity, Jaccard similarity, and Euclidean distance. These metrics are crucial for evaluating how closely textual data matches, by analyzing the occurrence and distribution of words and phrases within the texts [15].

The below table presents a side-by-side comparison between a single job vacancy's requirements and the qualifications of various candidates, alongside calculated similarity scores. These scores presumably indicate the degree of alignment between each candidate's experience and the job criteria, based on NLP text similarity algorithms.The similarity scores, likely generated by an advanced NLP model such as RoBERTa, provide a numerical representation of relevance, guiding the recruitment process in identifying the most suitable candidate for the position.

Vacancy Summary	About the Candidate	Similarity Score
Requires experience in data analytics, proficiency in SQL and Python, experience with Big Data tools, knowledge of machine learning, and must have leadership skills.	Skilled in data analysis with R.	0.74
	Proficient in SQL, Python, and data mining.	0.85
	Worked with Big Data technologies.	0.67
	Experienced in machine learning and AI.	0.74

Tab. 2.3: Text Similarity in Job Matching: Candidate and Vacancy Analysis

Euclidean Distance: This metric measures the straight-line distance between two points in Euclidean space, which can be extended to multidimensional data such as text vectors [18]. It is calculated using the formula:

$$Euclidean(A,B) = \sqrt{(y_B - y_A) + (y_B - y_A)} \tag{2.4}$$

Where A and B are two vectors in 2-dimensional space.

Cosine Similarity: Measures the cosine of the angle between two non-zero vectors (representations of text documents) within a multidimensional space. This metric evaluates how closely aligned the document vectors are in terms of direction, regardless of their magnitude. It is particularly effective for assessing similarity in high-dimensional data like text [18]. The cosine similarity is given by:

$$Cosine(A,B) = \frac{A.B}{|A||B|} \tag{2.5}$$

Where A and B are vector representations of the documents, and A . B denotes the dot product of the vectors.

Manhattan Distance: Also known as the Taxicab or City Block Distance, this metric sums the absolute differences of their Cartesian coordinates. It is used in various applications where the difference across multiple dimensions is to be accumulated [18]. The Manhattan distance between two points is calculated as:

$$Manhattan(A,B) = |x_B - y_A| + |x_B - y_A| \tag{2.6}$$

Where A and B are two vectors in 2-dimensional space.

Jaccard Similarity: Calculates the intersection over union of two sets of words. This method is straightforward and highly effective for cases where the presence or absence of words is more critical than the frequency of words [15].

Levenshtein Distance (Edit Distance): Determines the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other. This approach is valuable in spelling correction and matching closely related terms [15].

Hamming Distance: Measures the number of positions at which the corresponding symbols are different. It's used in scenarios where the strings are of the same length, helping to assess the degree of substitution needed to transform one string into another [15].

5.4.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) is a critical technique in Natural Language Processing (NLP) that involves identifying and classifying key information in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc [15]. This capability is fundamental to many applications, including job recommendation systems where understanding specific entities like job titles, skills, and educational qualifications is crucial.

NER helps to extract structured information from unstructured text, enabling recommendation systems to match job descriptions with candidate profiles more accurately. For instance, by recognizing and categorizing entities such as programming languages, certifications, or degree titles mentioned in resumes and job postings, NER systems can enhance the precision of candidate-job alignment.

The table below demonstrates the use of Named Entity Recognition (NER) to match job locations with candidate preferences. Using a BERT model, it assigns matching scores that reflect location relevance; for example, a perfect match for 'New York' is scored as 1. Minor spelling variations are well-tolerated, evident in a high score for a misspelled 'New Yor'. However, non-matching locations like 'San Francisco' receive low or zero scores, indicating a mismatch or a need for model adjustment for remote roles.

Vacancy Location	About the Candidate	Matching Score
New York	I am interested in finance roles in **New York**.	1.0
	Seeking finance opportunities in **New Yor**.	0.9
	Looking for tech jobs in **San Francisco**.	0.0
	Open to positions in **New York** or **Boston**.	0.95

Tab. 2.4: NER-Based Location Matching in Job Preferences

5.4.3 Keyword Extraction

Keyword extraction stands as an integral process within Natural Language Processing (NLP), focusing on the isolation of significant words or phrases from a vast text corpus that are vital in understanding the content and context. In the realm of job recommendation systems, keyword extraction is instrumental in discerning the essential qualifications and skills from job descriptions and resumes.

Several methods are employed for keyword extraction, including:

- **Term Frequency-Inverse Document Frequency (TF-IDF):** Identifies how important a word is to a document within a collection or corpus.

- **TextRank:** A graph-based ranking model for text processing which identifies keywords based on the number of times a word occurs and its relevance to the document.
- **Deep Learning Approaches:** Utilize complex models, often pre-trained on large datasets, to identify and weigh keywords according to context and semantic meaning.

The table below illustrates keyword matching between job requirements and candidate experiences using a Bert base extraction method. Candidates with direct experience in the required technologies and operating systems score higher, reflecting a stronger match. The simple comparison matching method quantifies the relevance of each candidate's profile to the vacancy.

Vacancy Summary	About the Candidate	Matching Score
Good familiarity with the **Windows** desktop environment and use of **Word**, **Excel**, IE, **Firefox** etc. are required. Perform execution and report results accurately.	I have experience with PyTorch, Tensorflow, and Keras.	0
	I have experience with different operating systems: **Windows**, Ubuntu, MacOS.	1
	I have experience in office applications: **Word**, **Excel**, PowerPoint, etc.	2

Tab. 2.5: Keyword Extraction for Matching Job Requirements with Candidate Experience

5.4.4 Text Summarization

Text Summarization in Natural Language Processing (NLP) is the process of distilling the most important information from a source text to produce a concise and coherent version. It plays a pivotal role in job recommendation systems by condensing lengthy job descriptions and candidate resumes into succinct overviews. The objective is to capture the essence of the information, enabling recruiters and job seekers to quickly grasp the key points without delving into the full text.

There are two primary types of text summarization:

1. **Extractive Summarization:** This approach identifies key sentences or phrases from the original text and extracts them to create a summary. It relies on algorithms to determine the weight and relevance of each sentence and compose a summary that represents the original content's salient points.
2. **Abstractive Summarization:** A more advanced technique that involves generating new sentences that may not necessarily appear in the source text. It interprets the main ideas using NLP and machine learning to produce a coherent summary that resembles human-generated narratives.

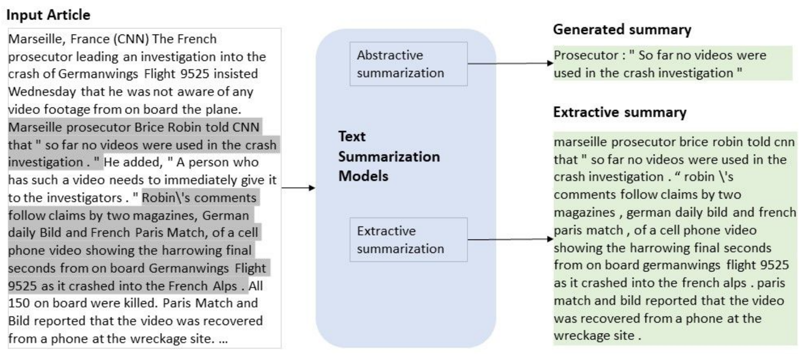

Input Article
Marseille, France (CNN) The French prosecutor leading an investigation into the crash of Germanwings Flight 9525 insisted Wednesday that he was not aware of any video footage from on board the plane. Marseille prosecutor Brice Robin told CNN that " so far no videos were used in the crash investigation . " He added, " A person who has such a video needs to immediately give it to the investigators . " Robin\'s comments follow claims by two magazines, German daily Bild and French Paris Match, of a cell phone video showing the harrowing final seconds from on board Germanwings Flight 9525 as it crashed into the French Alps . All 150 on board were killed. Paris Match and Bild reported that the video was recovered from a phone at the wreckage site. ...
Text Summarization Models
Abstractive summarization
Extractive summarization
Generated summary
Prosecutor : " So far no videos were used in the crash investigation "
Extractive summary
marseille prosecutor brice robin told cnn that " so far no videos were used in the crash investigation . " robin \'s comments follow claims by two magazines , german daily bild and french paris match , of a cell phone video showing the harrowing final seconds from on board germanwings flight 9525 as it crashed into the french alps . paris match and bild reported that the video was recovered from a phone at the wreckage site .

Fig. 2.14: Overview of Text Summarization Techniques in NLP.

6 Recommendations in recruitment

The traditional job search process, dominated by keyword-based searches and generic resumes, can be a frustrating and time-consuming endeavor for both candidates and employers. Fortunately, the rise of recommendation techniques is transforming the job hunting landscape by fostering smarter matching between candidates and suitable opportunities. Here's how these techniques are reshaping the recruitment scene:

Personalized Candidate Matching: Recommendation systems analyze a candidate's skills, experience, educational background, and potentially even social media activity to create a comprehensive skills profile. This profile is then compared to job descriptions, highlighting relevant requirements and experiences. This targeted approach eliminates irrelevant job postings that waste a candidate's time and allows them to focus on positions that truly align with their qualifications and career aspirations.

Skills-Based Recommendations: Beyond just matching job titles, recommendation systems can identify the specific skills and experiences most sought after by employers. These insights empower candidates to tailor their resumes and cover letters, emphasizing the skills required for the desired positions. Additionally, the system can recommend relevant courses, certifications, or training programs to help bridge any skill gaps identified in a candidate's profile.

Uncovering Hidden Job Opportunities: Recommendation systems can leverage a candidate's network and browsing behavior to discover hidden job opportunities that might not appear in traditional searches. For instance, the system might recommend positions at companies a candidate has shown interest in or connect them with former colleagues who may have valuable insights into potential job openings. This ability to unearth hidden gems expands the job search landscape and increases a candidate's chances of landing the perfect role.

Employer Branding and Targeted Advertising: For companies, recommendation systems offer a powerful tool to strengthen their employer brand and attract high-caliber candidates. By analyzing the profiles of successful employees, the system can identify the key skills and experiences that contribute to success within the company. This information can then be used to tailor job postings and employer branding initiatives to attract candidates who possess the right skill sets and cultural fit. Additionally, targeted advertising on social media platforms can be used to reach a pool of pre-qualified candidates based on their skills and interests, streamlining the recruitment process.

Building a Collaborative Ecosystem: Recommendation systems can foster a more collaborative job search ecosystem. By analyzing user behavior and feedback on job postings, the system can continuously learn and improve its matching algorithms. Additionally, candidate reviews of companies and interview experiences can be fed back into the system, providing valuable insights for both job seekers and employers in a transparent and dynamic way.

Overall, recommendation techniques are transforming the job search landscape from a random search to a strategic and personalized process. By empowering both candidates and employers with smarter matching and valuable insights, these techniques have the potential to revolutionize the way we find and fill jobs in the future.

7 Related works

7.1 Academic Research

- **Collaborative and Content-based Filtering:** Early research focused on collaborative and content-based filtering methods has laid the groundwork for modern recommendation systems. These studies highlighted the strengths and limitations of each approach, paving the way for hybrid models that combine the best of both worlds.

- **Matrix Factorization Techniques:** Significant academic work has been done on advanced matrix factorization models for improving recommendation accuracy, especially in sparse data environments common in job recommendation scenarios.

- **Deep Learning Applications:** Recent research has explored the use of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in enhancing the personalization and accuracy of recommendations.

7.2 Commercial Platforms

- **LinkedIn:** One of the pioneers in job recommendation, LinkedIn uses a combination of user data and machine learning algorithms to provide personalized job suggestions to its users.

- **Indeed:** As a major player in the job search engine market, Indeed utilizes complex algorithms to match resumes with job listings, offering a wide range of filtering and recommendation features to streamline the job search process.

- **Glassdoor:** Known for its company reviews and salary data, Glassdoor integrates these insights with job recommendations, helping users make more informed decisions about their career moves.

7.3 Emerging Technologies

- **AI and NLP in Job Matching:** Innovations in AI and NLP are being increasingly applied to understand deeper contextual meanings of job descriptions and resumes. This technology helps in better aligning candidate skills with job requirements beyond mere keyword matching.

- **Mobile Job Applications:** Platforms like ZipRecruiter and Monster have developed sophisticated mobile apps that use machine learning to adapt and personalize job recommendations based on user interactions and feedback.

8 Conclusion

Chapter 1 has provided a comprehensive foundation for understanding recommendation systems and their potential to revolutionize the recruitment landscape. We began by highlighting the challenges of traditional keyword-based job searches and the crucial role technology plays in modern recruitment.

The concept of recommendation systems was introduced, delving into their historical origins and exploring various types, including collaborative filtering, content-based filtering, and their hybrid counterparts. We then focused on specific techniques like matrix factorization and deep neural networks, which power the sophisticated matching algorithms utilized in modern systems.

The chapter further explored the diverse use cases and applications of recommendation systems beyond recruitment, showcasing their broader impact across various industries. We then highlighted the numerous benefits these systems offer, including improved efficiency, better candidate-job matching, and a more personalized job search experience. Finally, we examined how recommendation techniques can be applied to job finding, empowering candidates to discover relevant opportunities and showcase their qualifications more effectively.

This foundation paves the way for further exploration in subsequent chapters. By delving deeper into the specific algorithms, implementation details, and evaluation metrics associated with recommendation systems, we can gain a more nuanced understanding of their capabilities and limitations in the context of recruitment. We can then explore how these systems can be further fine-tuned and optimized to address specific challenges within the recruitment process and ultimately achieve the goal of connecting the right candidates with the right jobs.

In conclusion, Chapter 1 has established recommendation systems as a powerful tool with the potential to transform the recruitment industry. By fostering smarter matching, personalized experiences, and a more collaborative ecosystem, these systems hold the key to a future where both candidates and employers can navigate the job market with greater efficiency and success.

CHAPTER 3

Design and Implementation of Recommendation System

1 Introduction

In this chapter, we discuss the design and implementation of a content-based recommendation system aimed at enhancing user engagement by delivering personalized job recommendations. This system leverages user interaction data and job descriptions to identify and suggest job listings that align closely with user preferences and professional backgrounds. We detail the comprehensive approach taken from data acquisition and preprocessing to the application of sophisticated natural language processing techniques. The goal is to create a robust system that not only understands the intricate details of job data but also accurately predicts and recommends jobs that users are likely to find relevant and appealing. This chapter sets the stage for subsequent sections that delve into the system's architecture, the methodologies employed for data handling and analysis, and the metrics used to evaluate the system's performance.

2 Design

This section outlines the architecture and core principles of our content-based recommendation system, focusing on the dataset's characteristics, preprocessing methods, and the evaluation metrics employed. We detail how each design aspect contributes to enhancing the system's ability to deliver personalized and relevant job recommendations efficiently.

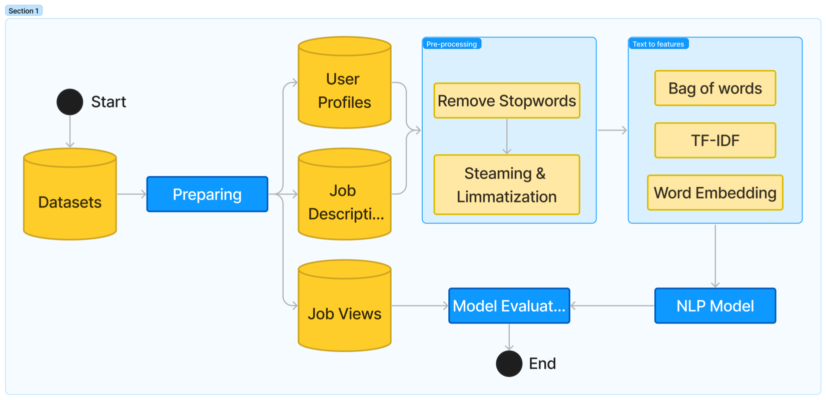


Fig. 3.1: Architecture of our content-based recommendation system

2.1 Dataset

2.1.1 Description

In the development of the content-based recommendation system, we utilized four distinct datasets from Kaggle [19], each providing essential data elements necessary for constructing a comprehensive profile of both jobs and applicants. These datasets include:

Dataset 1: (Combined_Jobs_Final.csv) It contains detailed information about job listings, including job titles, descriptions, company details, and geographic location. This dataset forms the core of the job recommendation system by providing the data needed for job content analysis and feature extraction.

Dataset 2: (Experience.csv) It catalogs the previous job experiences of applicants, listing their former positions, employers, job locations, and descriptions. This dataset is crucial for understanding applicants' professional backgrounds and preferences, aiding in the personalization of job recommendations.

Dataset 3: (Positions_Of_Interest.csv) It contains information about the job positions applicants have expressed interest in. This dataset is instrumental in aligning job recommendations with the explicit preferences of the users.

Dataset 4: (Job_Views.csv) This dataset tracks which jobs applicants have viewed, providing insights into applicant interests and engagement. It includes data on the applicant ID, job ID, and timestamps of views, essential for analyzing interaction patterns.

2.1.2 Structure

In this section, we detail the structure of the datasets utilized in our recommendation system. Each dataset comprises various fields that capture significant attributes relevant to job seekers and job postings. The datasets and their structures are outlined in table below:

Dataset	Fields
Positions Of Interest	• **Applicant.ID:** Unique identifier for each applicant • **Position.Of.Interest:** Type or title of the job position the applicant is interested in • **+9 other:** Additional attributes not specified here
Experience	• **Applicant.ID:** Unique identifier for each applicant • **Position.Name:** Name of the position held by the applicant • **+11 other:** Additional attributes not specified here
Job Views	• **Applicant.ID:** Unique identifier for each applicant • **Job.ID:** Unique identifier for each job viewed • **Position:** Specific position of the job listing • **Company:** Name of the company offering the job • **City:** City where the job is located • **+9 other:** Additional attributes not specified here

Continued on next page

Tab. 3.1 – continued from previous page

Dataset	Fields
Jobs	• **Job.ID:** Unique identifier for each job listing • **Slug:** A slug is a human-readable, unique identifier, used to identify a resource instead of a less human-readable identifier like an id. • **Title:** Title of the job listing • **Position:** Specific position title within the company • **Company:** Name of the company offering the job • **City:** City where the job is located • **Employment.Type:** Type of employment (e.g., Full-time, Part-time) • **Education.Required:** Educational requirements for the job • **Job.Description:** Description of the job responsibilities • **+14 other:** Additional attributes not specified here

Tab. 3.1: Structure of used datasets

The figure below illustrates the structure and relationships of four datasets used in a job recommendation system: Positions of Interest, Experience, Job Views, and Jobs.

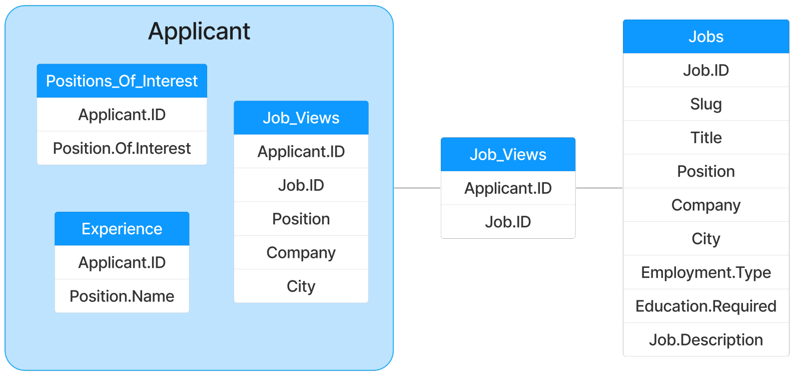


Fig. 3.2: Structure and relationships of datasets

2.1.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a fundamental process in data analysis that involves summarizing main characteristics of datasets, often through visual methods. The primary objective of EDA is to uncover patterns, spot anomalies, check assumptions, and test hypotheses about the data under analysis. This crucial step provides deep insights that inform subsequent data preparation and modeling phases. EDA aims to optimize the analytical models and enhance decision-making processes.

The bar chart below illustrates the distribution of job listings across various industries within the dataset. It's evident that the "Food and Beverages" industry dominates, accounting for the majority of job postings, followed by industries such as "Office Administration," "Transportation," and "Retail." The "Care Giving" sector has the fewest listings, indicating a potential underrepresentation in the dataset.

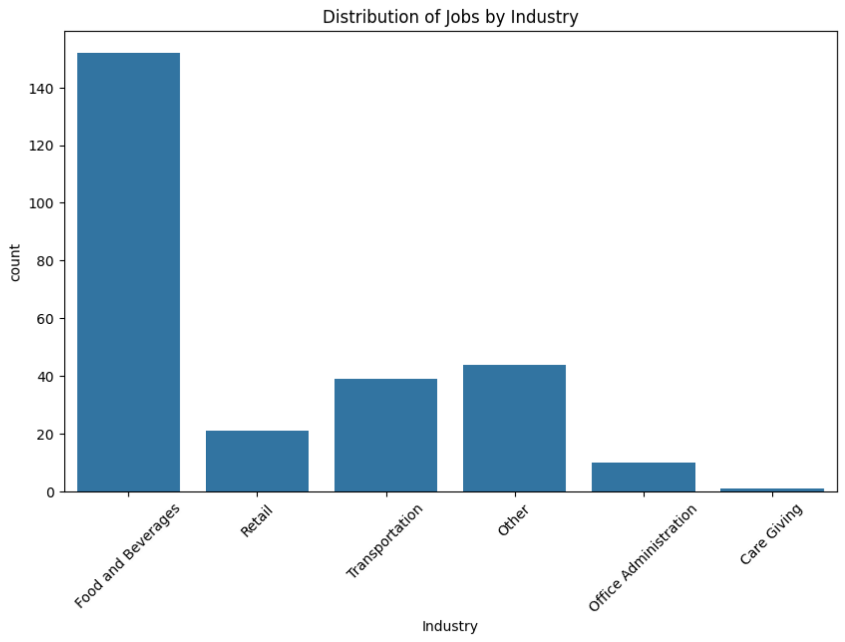


Fig. 3.3: Distribution of jobs by industry

The word cloud displayed illustrates the most frequently occurring terms in job descriptions from the dataset. Prominent words like "Home," "Health," "Sales," "Service," "Associate," "Customer," and "Clerk" indicate key focus areas in job roles and sectors. This visualization provides a snapshot of the skills and positions that are currently in high demand across various industries.

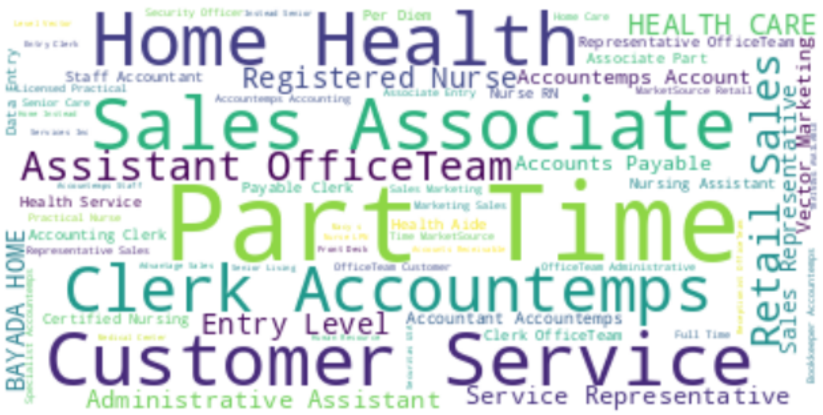


Fig. 3.4: Word Cloud of frequently occurring terms in job descriptions

The histogram above represents the distribution of the number of job views per user in the dataset. It clearly shows that the majority of users view between 0 to 10 jobs, with a sharp decrease as the number of job views increases.

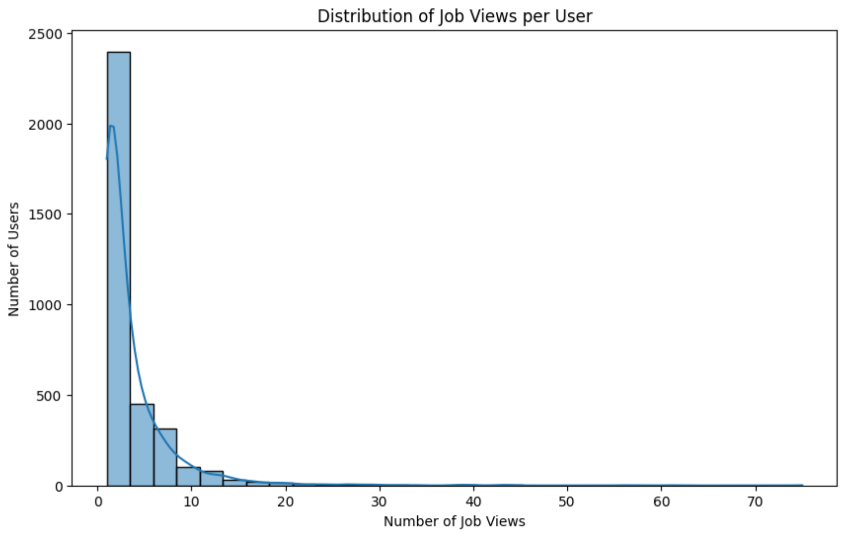


Fig. 3.5: Distribution of job views per User

2.2 Preparing & Preprocessing

The preparation and preprocessing of data are critical steps in our recommendation system's pipeline, ensuring that the input data is clean, structured, and suitable for further analysis and model building. The process is visualized in the accompanying diagram, illustrating how multiple datasets are integrated and refined into structured forms suitable for processing.

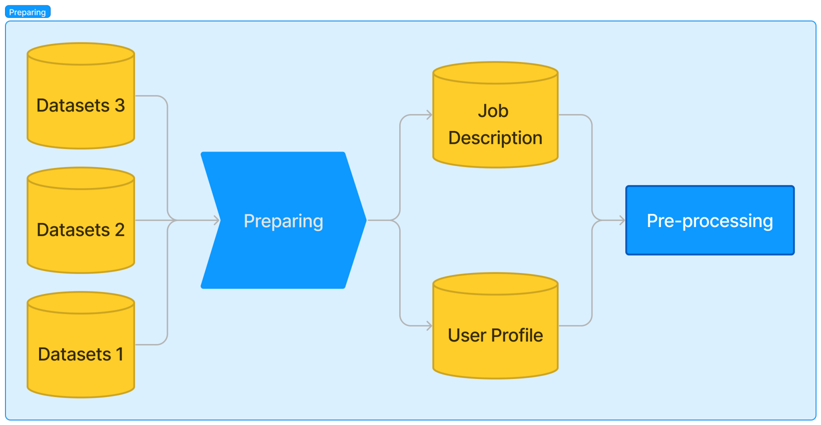


Fig. 3.6: Data preparation and preprocessing workflow

2.2.1 Preparing

In this initial phase, data from multiple sources is consolidated and cleaned to create a unified dataset. This preparation is essential for aligning the data into a format suitable for detailed analysis and subsequent processing.

- **Data Cleaning**

  We perform extensive data cleaning which involves:

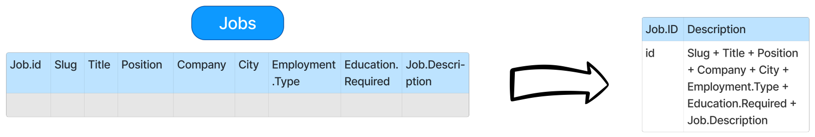
  Removing Duplicates: Ensuring each data entry is unique to prevent skewed analysis.

  Handling Missing Values: Filling in or omitting missing values to improve data quality.

  Data Verification: Checking for data consistency and correctness across different sources.

- **Fetching Job Description**

  Job descriptions are extracted and organized to highlight key information such as title, description, and position which are critical for accurate job matching.



Job.id	Slug	Title	Position	Company	City	Employment .Type	Education. Required	Job.Descri- ption

Job.ID	Description
id	Slug + Title + Position + Company + City + Employment.Type + Education.Required + Job.Description

Fig. 3.7: Constructing job description form datasets

- **Fetching User Profile**

User profiles are constructed using demographic data, historical interaction data, and preferences to enable personalized job recommendations tailored to individual needs.

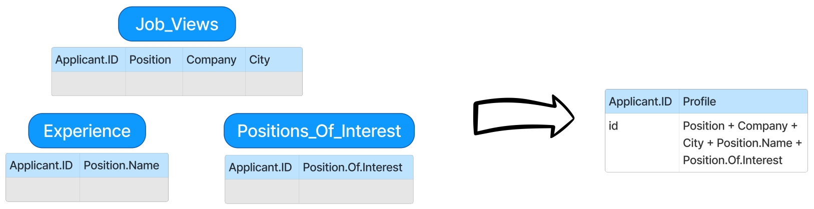


Fig. 3.8: Constructing user profile form datasets

2.2.2 Preprocessing

The preprocessing stage refines the text data through several techniques to enhance the machine readability and analytical value of the content.

- **Removing Stopwords**

  Commonly used words (such as "the", "is", "at") which do not add significant meaning to the text are removed to focus on more meaningful words.

- **Deleting Special Characters**

  Special characters and numbers are removed from the text to prevent any analytical distortions and focus on the textual content.

- **Converting to Lowercase**

  All text data is converted to lowercase to maintain uniformity and avoid discrepancies in text processing.

- **Stemming & Lemmatization**

  Words are reduced to their base or root form, smoothing out variations in the data and improving the consistency of textual analysis.

- **Tokenization**

  Text data is split into individual words or tokens, making it easier to apply natural language processing techniques and analyze the data effectively.

After these steps, the data is well-prepared and preprocessed, formatted into two primary schemas: one for job descriptions and another for user profiles, ready to be utilized in the recommendation engine for further analysis and model training.

2.3 Evaluation metrics

In our recommendation system, we utilize classification accuracy measures such as Precision, Recall, and the F1 Score to evaluate the effectiveness of our recommendations. These metrics are especially pertinent to systems that make binary decisions about user-item pairs, i.e., whether to recommend or not recommend a particular item to a user.

Precision and Recall are applied by transforming the recommendation system's output from a continuous or multi-class scale into a binary scale. In this context, items predicted to be of interest are labeled as "Recommend," and those deemed not of interest are labeled "Do not recommend." This binary categorization allows us to directly measure the system's accuracy in predicting relevant items. Each item can be either relevant or irrelevant to the user. We get, therefore, the following matrix:

Vacancy Summary	**Recommended**	**Not Recommended**	**Total**
Relevant	RR	RN	R = RR + RN
Not Relevant	FP	NN	IR = F P + NN
Total	REC = RR + F P	NREC = RN + NN	N = R + IR = REC + NREC

Tab. 3.2: Matrix for recommendation system evaluation

- **Precision:** Precision measures the proportion of recommended items that are relevant. It is calculated as

$$\text{Precision} = \frac{RR}{RR + FP} = \frac{RR}{REC} \tag{3.1}$$

- **Recall:** Recall measures the proportion of relevant items that have been recommended. It is calculated as

$$\text{Recall} = \frac{RR}{RR + RN} = \frac{RR}{R} \tag{3.2}$$

- **F-measure (F1 Score):** The F-measure, or F1 Score, is the harmonic mean of precision and recall, providing a single metric that balances both the precision and the recall. It is calculated as

$$F1 = 2\frac{\text{Precision}\text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

3 Implementation

3.1 Frameworks, tools and libraries

- **Python:**

Python is a versatile and widely-used programming language known for its readability and broad support for scientific computing and data analysis. It is popular in data science due to its powerful libraries and active community.

Fig. 3.9: Python logo

- **Google Colab:**

  Google Colab is a cloud-based platform that allows users to write and execute Python code through the browser. It is equipped with free access to GPUs and TPUs which makes it ideal for machine learning and data analysis without requiring local setup.

Fig. 3.10: Google colab logo

- **Pandas:**

  Pandas is a library in Python that provides data structures and data analysis tools. It excels in handling and manipulating structured data, such as tables, with its DataFrame object.

Fig. 3.11: Pandas logo

- **NumPy:**

  NumPy is a fundamental package for scientific computing in Python. It supports large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Fig. 3.12: NumPy logo

- **Matplotlib:**

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Fig. 3.13: Matplotlib logo

- **Kaggle:**

  Kaggle is a platform for data science competitions and collaborative projects. It provides datasets, a cloud-based work environment, and a community for data scientists and machine learning practitioners to engage and collaborate.

Fig. 3.14: Kaggle logo

3.2 Data Loading and Preparation

The following code loads and prepares the job and user data for a content-based recommendation system. It combines relevant columns to create comprehensive job descriptions and user profiles by aggregating positions of interest and experiences. This preparation ensures clean and structured data, ready for further processing and analysis. **Load Data**

```python
import pandas as pd

# Load datasets
jobs_df = pd.read_csv('drive/MyDrive/Colab Notebooks/input/Combined_Jobs_Final.csv')
job_views_df = pd.read_csv('drive/MyDrive/Colab Notebooks/input/Job_Views.csv')
experience_df = pd.read_csv('drive/MyDrive/Colab Notebooks/input/Experience.csv')
position_of_interest_df = pd.read_csv('drive/MyDrive/Colab Notebooks/input/Positions_Of_Interest.csv')
```

Listing 3.1: Loading datasets

Job Description

```python
# Select relevant columns from the jobs DataFrame that will be used to create the job description
```

```python
jobs_df = jobs_df[['Job.ID', 'Title', 'Position', 'Company', 'City', 'State.Name', 'Employment.Type', 'Education.Required', 'Job.Description']]

# Remove rows with any missing values to ensure clean data
jobs_df = jobs_df.dropna()

# Combine the selected columns into a single 'Description' column
jobs_df['Description'] = jobs_df['Title'] + ' ' + jobs_df['Position'] + ' ' + jobs_df['Company'] + ' ' + jobs_df['City'] + ' ' + jobs_df['State.Name'] + ' ' + jobs_df['Employment.Type'] + ' ' + jobs_df['Education.Required'] + ' ' + jobs_df['Job.Description']

# Create a new DataFrame with only 'Job.ID' and the newly created 'Description' column
job_description_df = jobs_df[['Job.ID', 'Description']]

# Limit the job descriptions to the first 5000 entries to reduce computational load
job_description_df = job_description_df[:5000]
```

Listing 3.2: Constructing job descriptions

User Profile

```python
# Combine relevant columns to create the 'Experience' column for each applicant
# This concatenates the position name, city, and job description into a single string
experience_df['Experience'] = experience_df['Position.Name'] + ' ' + experience_df['City'] + ' ' + experience_df['Job.Description']

# Select 'Applicant.ID' and 'Experience' columns and remove rows with any missing values
experience_df = experience_df[['Applicant.ID', 'Experience']].dropna()

# Group by 'Applicant.ID' and concatenate all experiences into a single string for each applicant
experience_df = experience_df.groupby('Applicant.ID', sort=False)['Experience'].apply(' '.join).reset_index()

# Select 'Applicant.ID' and 'Position.Of.Interest' columns and remove rows with any missing values
position_of_interest_df = position_of_interest_df[['Applicant.ID', 'Position.Of.Interest']].dropna()

# Group by 'Applicant.ID' and concatenate all positions of interest into a single string for each applicant
position_of_interest_df = position_of_interest_df.groupby('Applicant.ID', sort=False)['Position.Of.Interest'].apply(' '.join).reset_index()

# Merge the 'Position.Of.Interest' data with the 'Experience' data on 'Applicant.ID'
# This creates a combined user profile containing both positions of interest and experiences
```

```
user_profile_df = position_of_interest_df.merge(experience_df[['Applicant.ID', 'Experience']], on='Applicant.ID', how='left')

# Concatenate the 'Position.Of.Interest' and 'Experience' columns to form the final 'Profile' for each applicant
user_profile_df['Profile'] = user_profile_df['Position.Of.Interest'] + ' ' + user_profile_df['Experience'].fillna('')

# Select 'Applicant.ID' and 'Profile' columns and remove rows with any missing values
user_profile_df = user_profile_df[['Applicant.ID', 'Profile']].dropna()

# Limit the user profiles to the first 5000 entries to reduce computational load
user_profile_df = user_profile_df[:5000]
```

Listing 3.3: Constructing user profiles

3.3 Data Pre-processing

The following code performs text pre-processing to clean and normalize the job descriptions and user profiles. It involves converting text to lowercase, removing special characters, eliminating stopwords, and applying stemming and lemmatization. These steps prepare the text data for efficient and accurate feature extraction.

Import Necessary Libraries

```
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk import download
```

Listing 3.4: Import necessary libraries for text preprocessing

Define the Preprocessing Function

```
# Initialize stemmer and lemmatizer
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Define preprocessing function
def preprocess_text(text):
    text = text.lower()  # Convert text to lowercase
    text = re.sub(r'\W', ' ', text)  # Remove special characters
    tokens = text.split()  # Split text into tokens
    tokens = [word for word in tokens if word not in stop_words]  # Remove stopwords
    tokens = [stemmer.stem(word) for word in tokens]  # Apply stemming
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Apply lemmatization
    return ' '.join(tokens)  # Rejoin tokens into a single string
```

Listing 3.5: Text preprocessing function for NLP

Apply Preprocessing

```python
# Apply preprocessing to job descriptions
job_description_df['Description'] = job_description_df['Description'].
    apply(preprocess_text)

# Apply preprocessing to user profiles
user_profile_df['Profile'] = user_profile_df['Profile'].apply(
    preprocess_text)
```

Listing 3.6: Applying Text Preprocessing

3.4 Text to Features

The following code extracts features from the text data using different techniques: TF-IDF, Bag of Words, and Word Embeddings. Each technique is implemented as a function to transform the job descriptions and user profiles into numerical representations suitable for machine learning algorithms.

3.4.1 TF-IDF

TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It transforms text data into numerical features by considering the frequency of words and their inverse document frequency.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer()

def extract_features_tfidf(job_description_df, user_profile_df):
    # Fit and transform the job descriptions
    tfidf_matrix_jobs = tfidf_vectorizer.fit_transform(
    job_description_df['Description'])

    # Transform the user profiles
    tfidf_matrix_profiles = tfidf_vectorizer.transform(user_profile_df['
    Profile'])

    return tfidf_matrix_jobs, tfidf_matrix_profiles
```

Listing 3.7: TF-IDF function

3.4.2 Bag of Words

Bag of Words is a simple technique that transforms text data into numerical features by counting the occurrence of each word in the text. It ignores the order of words and treats each word independently.

```python
from sklearn.feature_extraction.text import CountVectorizer

# Initialize Bag of Words vectorizer
bow_vectorizer = CountVectorizer()
```

```python
def extract_features_bow(job_description_df, user_profile_df):
    # Fit and transform the job descriptions
    bow_matrix_jobs = bow_vectorizer.fit_transform(job_description_df['Description'])

    # Transform the user profiles
    bow_matrix_profiles = bow_vectorizer.transform(user_profile_df['Profile'])

    return bow_matrix_jobs, bow_matrix_profiles
```

Listing 3.8: Bag of Words function

3.4.3 Words Embeddings

Word Embeddings represent words as dense vectors in a continuous vector space. These vectors capture the semantic meaning of words. The Universal Sentence Encoder (USE) is used to obtain embeddings for the entire sentences or documents.

```python
import tensorflow_hub as hub

# Load Universal Sentence Encoder
embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")

def extract_features_embeddings(job_description_df, user_profile_df):
    # Convert text to embeddings
    job_embeddings = embed(job_description_df['Description'].tolist())
    profile_embeddings = embed(user_profile_df['Profile'].tolist())

    return job_embeddings, profile_embeddings
```

Listing 3.9: Words embeddings function

3.4.4 Text to feature function

The text_to_feature function converts job descriptions and user profiles into numerical feature representations using the specified technique (TF-IDF, Bag of Words, or Word Embeddings). This enables subsequent analysis and similarity calculations for personalized job recommendations.

```python
def text_to_feature(job_description_df, user_profile_df, technique="TF-IDF"):
    # Initialize feature matrices
    matrix_jobs, matrix_profiles = None, None
    # Extract features based on the selected technique
    if technique == "TF-IDF":
        matrix_jobs, matrix_profiles = extract_features_tfidf(job_description_df, user_profile_df)
    elif technique == "BoW":
        matrix_jobs, matrix_profiles = extract_features_bow(job_description_df, user_profile_df)
    elif technique == "Embeddings":
        matrix_jobs, matrix_profiles = extract_features_embeddings(job_description_df, user_profile_df)
    else:
        raise ValueError("Invalid technique specified. Choose from 'TF-IDF', 'BoW', or 'Embeddings'.")
    return matrix_jobs, matrix_profiles
```

Listing 3.10: Text to feature function

3.5 NLP model (Cosine Similarity)

Cosine similarity is a metric used to measure how similar two items are, irrespective of their size. In the context of job recommendations, it allows us to compare user profiles with job descriptions to find the best matches.

The following code uses cosine similarity to measure the similarity between user profiles and job descriptions. Based on this similarity, it generates top-N job recommendations for each user or for each job. The feature extraction technique can be switched between TF-IDF, Bag of Words, or Word Embeddings.

3.5.1 Jobs Recommendations

To generate job recommendations, we leverage cosine similarity to compare user profiles against job descriptions. This method identifies the top job listings that align closely with each user's profile.

```
from sklearn.metrics.pairwise import cosine_similarity
def generate_recommended_jobs(job_description_df, user_profile_df, technique= "TF-IDF"):
    matrix_jobs, matrix_profiles = text_to_feature(job_description_df, user_profile_df, technique)
    # Compute cosine similarity between user profiles and job descriptions
    cosine_similarities = cosine_similarity(matrix_profiles, matrix_jobs)
    # Generate top-N recommendations for each user
    top_n = 5  # Number of top recommendations to consider
    recommendations = {}
    for idx, similarities in enumerate(cosine_similarities):
        top_indices = similarities.argsort()[-top_n:][::-1]
        recommendations[user_profile_df.iloc[idx]['Applicant.ID']] = job_description_df.iloc[top_indices]['Job.ID'].values
    return recommendations
```

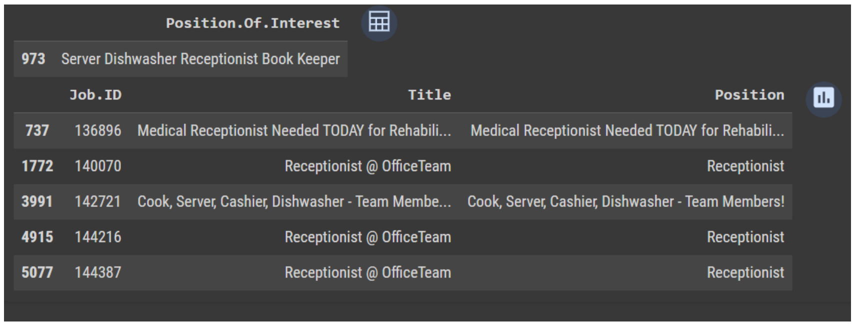
Listing 3.11: Jobs recommendation

Testing: To validate the job recommendation system, we generate recommendations for a specific user and display the results. This involves checking the recommended jobs against the user's positions of interest.

```
recommendations = generate_recommended_jobs(job_description_df, user_profile_df)
```

```python
display(position_of_interest_df[position_of_interest_df['Applicant.ID']
    == 3149][['Position.Of.Interest']])
jobs_df[jobs_df['Job.ID'].isin(recommendations[3149])][['Job.ID', 'Title
    ', 'Position']]
```

Listing 3.12: Testing jobs recommendation



```
     Position.Of.Interest
973  Server Dishwasher Receptionist Book Keeper

      Job.ID  Title                                          Position
737   136896  Medical Receptionist Needed TODAY for Rehabili...  Medical Receptionist Needed TODAY for Rehabili...
1772  140070  Receptionist @ OfficeTeam                      Receptionist
3991  142721  Cook, Server, Cashier, Dishwasher - Team Membe...  Cook, Server, Cashier, Dishwasher - Team Members!
4915  144216  Receptionist @ OfficeTeam                      Receptionist
5077  144387  Receptionist @ OfficeTeam                      Receptionist
```

Fig. 3.15: Recommended jobs for applicant

3.5.2 Applicants Recommender

Similar to job recommendations, we can also recommend applicants for specific job listings. This process uses cosine similarity to find user profiles that match the job descriptions, suggesting the best candidates for each job.

```python
def generate_recommended_applicant(job_description_df, user_profile_df,
    technique= "TF-IDF"):
     matrix_jobs, matrix_profiles = text_to_feature(job_description_df,
    user_profile_df, technique)
     # Compute cosine similarity between user profiles and job
    descriptions
     cosine_similarities = cosine_similarity(matrix_jobs, matrix_profiles
    )
     # Generate top-N recommendations for each user
     top_n = 5  # Number of top recommendations to consider
     recommendations = {}
     for idx, similarities in enumerate(cosine_similarities):
         top_indices = similarities.argsort()[-top_n:][::-1]
         recommendations[job_description_df.iloc[idx]['Job.ID']] =
    user_profile_df.iloc[top_indices]['Applicant.ID'].values
     return recommendations
```

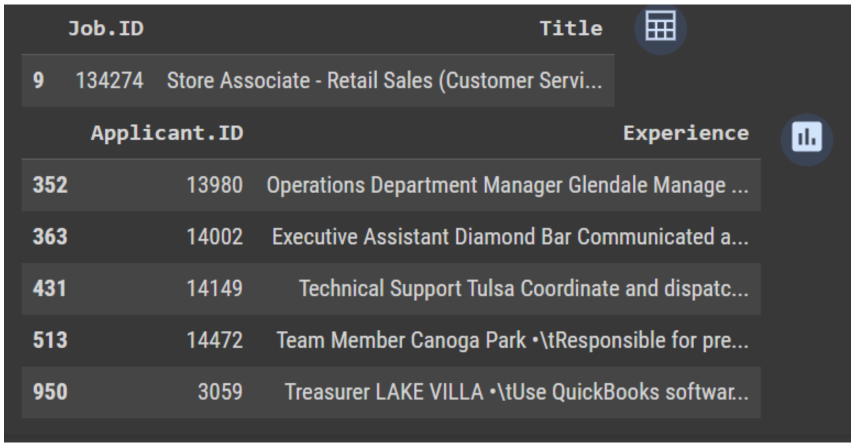
Listing 3.13: Applicants recommendation

Testing: To validate the applicant recommendation system, we generate recommendations for a specific job and display the results. This involves checking the recommended applicants against the job's requirements.

```python
recommendations_for_jobs = generate_recommended_applicant(
    job_description_df, user_profile_df)
print(job_description_df[job_description_df['Job.ID'] == 134273])
```

```
user_profile_df[user_profile_df['Applicant.ID'].isin(
    recommendations_for_jobs[134273])]
```

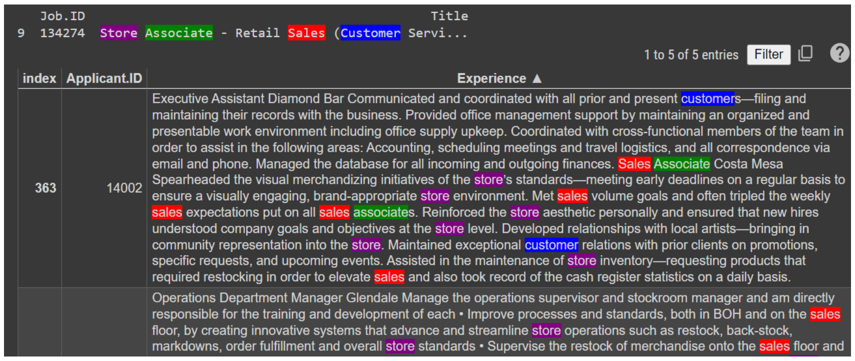
Listing 3.14: Testing applicants recommendation



```
   Job.ID                                      Title
9  134274  Store Associate - Retail Sales (Customer Servi...

     Applicant.ID                                      Experience
352         13980  Operations Department Manager Glendale Manage ...
363         14002    Executive Assistant Diamond Bar Communicated a...
431         14149       Technical Support Tulsa Coordinate and dispatc...
513         14472   Team Member Canoga Park •\tResponsible for pre...
950          3059     Treasurer LAKE VILLA •\tUse QuickBooks softwar...
```

Fig. 3.16: Recommended applicants for job

The recommendations initially appear to be inappropriate for the required job because the full text was not displayed. However, upon printing the entire text, we find that there is a match, as shown in the figure below.



```
  Job.ID                                      Title
9 134274  Store Associate - Retail Sales (Customer Servi...
```

1 to 5 of 5 entries Filter

index	Applicant.ID	Experience ▲
363	14002	Executive Assistant Diamond Bar Communicated and coordinated with all prior and present customers—filing and maintaining their records with the business. Provided office management support by maintaining an organized and presentable work environment including office supply upkeep. Coordinated with cross-functional members of the team in order to assist in the following areas: Accounting, scheduling meetings and travel logistics, and all correspondence via email and phone. Managed the database for all incoming and outgoing finances. Sales Associate Costa Mesa Spearheaded the visual merchandizing initiatives of the store's standards—meeting early deadlines on a regular basis to ensure a visually engaging, brand-appropriate store environment. Met sales volume goals and often tripled the weekly sales expectations put on all sales associates. Reinforced the store aesthetic personally and ensured that new hires understood company goals and objectives at the store level. Developed relationships with local artists—bringing in community representation into the store. Maintained exceptional customer relations with prior clients on promotions, specific requests, and upcoming events. Assisted in the maintenance of store inventory—requesting products that required restocking in order to elevate sales and also took record of the cash register statistics on a daily basis.
		Operations Department Manager Glendale Manage the operations supervisor and stockroom manager and am directly responsible for the training and development of each • Improve processes and standards, both in BOH and on the sales floor, by creating innovative systems that advance and streamline store operations such as restock, back-stock, markdowns, order fulfillment and overall store standards • Supervise the restock of merchandise onto the sales floor and

Fig. 3.17: Detailed experience of recommended applicants for job

This section demonstrates how cosine similarity can be effectively used to generate both job and applicant recommendations. By testing the system with real data, we can ensure its reliability and accuracy in making relevant recommendations.

4 Testing

To test the effectiveness of the recommendations made by our system, we can evaluate how well the recommended job IDs match the job IDs that users have actually interacted with, as recorded in our user interaction dataset. This approach involves comparing the recommended jobs for a user against their historical job interactions, assessing the precision, recall, and F1 score of the recommendations.

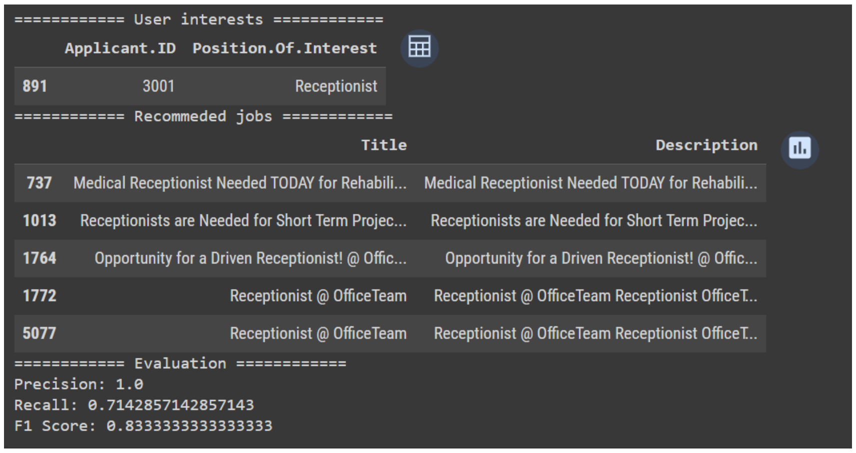
We focus solely on job recommendations for users, as we are unable to test applicant recommendations for jobs. This limitation is due to the lack of ground truth data necessary to evaluate the results of applicant recommendations.

Here's how to set up and conduct this evaluation:

```python
from sklearn.metrics import precision_score, recall_score, f1_score

# Generate recommendations for all users
recommendations = generate_recommended_jobs(job_description_df,
    user_profile_df, technique="TF-IDF")

# Evaluates the recommendations for a specific user by calculating
    precision, recall, and F1 score.
def evaluate_jobs_recommendations(recommendations, user_interactions_df,
     user_id):
    # Get the recommended jobs for the user
    recommended_jobs = recommendations.get(user_id, [])

    # Filter the jobs_df to get the information of the recommended jobs
    recommended_jobs_info = jobs_df[jobs_df['Job.ID'].isin(
    recommended_jobs)][['Title', 'Description']]

    # Get the actual jobs the user interacted with
    actual_jobs = user_interactions_df[user_interactions_df['Applicant.
    ID'] == user_id]['Job.ID'].values

    # Create ground truth and prediction arrays
    ground_truth = []
    predictions = []

    # Iterate over all jobs in the dataset to create binary labels
    all_jobs = job_description_df['Job.ID'].values
    for job_id in all_jobs:
        ground_truth.append(1 if job_id in actual_jobs else 0)
        predictions.append(1 if job_id in recommended_jobs else 0)

    # Calculate evaluation metrics
    precision = precision_score(ground_truth, predictions)
    recall = recall_score(ground_truth, predictions)
    f1 = f1_score(ground_truth, predictions)

    return recommended_jobs_info, precision, recall, f1

# Example user ID to evaluate
user_id_to_evaluate = 3001
positions = position_of_interest_df[position_of_interest_df['Applicant.
    ID'] == user_id_to_evaluate][['Applicant.ID', 'Position.Of.Interest'
    ]]
# Evaluate the recommendations for the specific user
recommended_jobs, precision, recall, f1 = evaluate_jobs_recommendations(
    recommendations, user_inter, user_id_to_evaluate)
```

```python

print("============ User interests ============")
display(positions.head())
print("============ Recommeded jobs ============")
display(recommended_jobs.head())
print("============ Evaluation ============")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

Listing 3.15: Testing system



```
=========== User interests ===========
     Applicant.ID  Position.Of.Interest
891          3001          Receptionist
=========== Recommeded jobs ===========
                                                  Title                                        Description
737   Medical Receptionist Needed TODAY for Rehabili...  Medical Receptionist Needed TODAY for Rehabili...
1013  Receptionists are Needed for Short Term Projec...  Receptionists are Needed for Short Term Projec...
1764     Opportunity for a Driven Receptionist! @ Offic...     Opportunity for a Driven Receptionist! @ Offic...
1772                          Receptionist @ OfficeTeam  Receptionist @ OfficeTeam Receptionist OfficeT...
5077                          Receptionist @ OfficeTeam  Receptionist @ OfficeTeam Receptionist OfficeT...
=========== Evaluation ===========
Precision: 1.0
Recall: 0.7142857142857143
F1 Score: 0.8333333333333333
```

Fig. 3.18: Testing result

5 Conclusion

In this chapter, we presented the structure of our datasets and described our system design, detailing the exact processes involved in the preprocessing phase. We also discussed the tools, libraries, and frameworks utilized, particularly focusing on natural language processing (NLP) techniques for analyzing and interpreting job-related text data. The implementation of our recommendation system was thoroughly explained, highlighting how NLP facilitates the creation of targeted job recommendations. In the next chapter, we will explore various experimentations and analyze the results obtained from our system.

CHAPTER 4

Experiments and Results

1 Introduction

In this chapter, we systematically present the methodology, setup, and outcomes of experiments conducted to evaluate the performance of our recommendation system. We detail the experimental environment, define the criteria for performance evaluation, and assess the effectiveness of our recommendations. This chapter aims to transparently quantify the impact and efficiency of the system, providing a comprehensive analysis of its operational capabilities and results.

2 Experiment environment

The table below delineates the comprehensive setup of the experimental environment used to test and evaluate our job recommendation system.

Component	**Specification**
Hardware	CPU: Intel i7-9700K, RAM: 16GB, GPU: NVIDIA GTX 1080
Software	Operating System: Windows 10, Python 3.8
Development Tools	Jupyter Notebook
Libraries and Frameworks	Pandas, NumPy, scikit-learn, NLTK, Matplotlib
Datasets	Combined_Jobs_Final.csv, Job_Views.csv, Experience.csv, Positions_Of_Interest.csv

Continued on next page

Tab. 4.1 – continued from previous page

Component	Specification
Algorithms	TF-IDF, Bag of Words, Embeddings, Cosine Similarity
Performance Metrics	Precision, Recall, F1 Score

Tab. 4.1: Experimental environment specifications

3 Evaluation of recommendations

The experiments were designed to evaluate the performance of the recommendation system using different feature extraction techniques and similarity measures. We conducted experiments using TF-IDF, Bag of Words, and Word Embeddings to extract features from the job descriptions and user profiles. We then calculated the cosine similarity between these features to generate job recommendations.

The evaluation was focused solely on job recommendations for users and did not include applicant recommendations for jobs. This is because we do not have ground truth data to evaluate the results for applicant recommendations.

3.1 Result

The results of the experiments are presented in the following table, where we analyze the performance of each feature extraction technique and discuss the implications of our findings. Each technique's precision, recall, and F1 score are compared to determine the most effective method for our recommendation system.

	TF-IDF	**Bag of Words**	**Embeddings**
Precision	0.86	0.50	0.62
Recall	0.46	0.23	0.38
F1 Score	0.60	0.32	0.48

Tab. 4.2: Table of results for different models

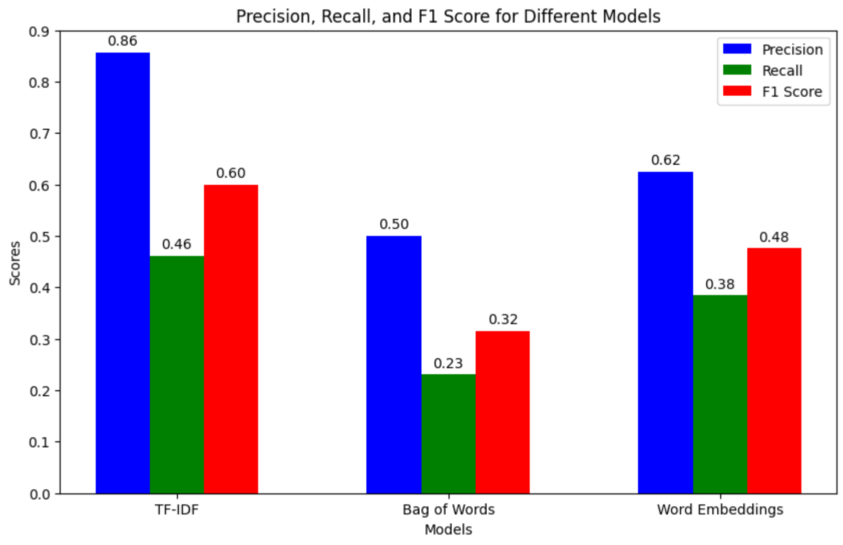


Fig. 4.1: Precision, Recall and F1 score for different models

3.2 Discussion

The results demonstrate that the TF-IDF technique achieved the highest precision and F1 score, indicating that it is the most effective method for generating relevant job recommendations. The Bag of Words method had the lowest performance across all metrics, suggesting that it may not be suitable for this application. Word Embeddings showed a balanced performance, making it a viable alternative to TF-IDF.

There are a couple of reasons to explain why TF-IDF was superior:

1. TF-IDF focuses on the importance of terms within a document relative to a collection of documents, effectively highlighting the most relevant terms without the additional complexity. This simplicity helps in capturing the core content of job descriptions and user profiles, leading to more accurate matches based on essential keywords and phrases.

2. The Bag of Words (BoW) technique performed poorly because it treats each word independently, losing context and semantic meaning, and results in a sparse feature matrix that fails to capture relationships between words, leading to fragmented representations and reduced accuracy.

3. A word embedding is a much more complex word representation and carries much more hidden information. In our case, most of that information is unnecessary and creates false patterns in our model. This noise can detract from the algorithm's ability to make accurate predictions, as it may overfit to irrelevant features present in the embeddings.

4 Conclusion

In this chapter, we conducted comprehensive experiments to evaluate the performance of our content-based recommendation system. By comparing different feature extraction techniques—TF-IDF, Bag of Words, and Word Embeddings—we identified significant differences in their effectiveness.

These findings highlight the importance of selecting appropriate feature extraction techniques to enhance the relevance and accuracy of job recommendations. Leveraging NLP techniques allows the system to better analyze and interpret job-related text data, resulting in more precise and targeted recommendations.

CHAPTER 5

Platform Development and Recommendation System Integration

1 Introduction

This chapter delves into the creation of the AI-powered job recommendation platform, detailing the end-to-end development process. It covers the system architecture, frontend and backend development, and the integration of the recommendation system. Emphasis is placed on the design choices, technologies used, and the implementation strategies to build a functional and efficient platform.

2 System Architecture

2.1 Architecture of the platform

This architecture represents a system designed to integrate various technologies to deliver a robust application comprising a client interface, server-side processing, and database management. The system includes a Flutter client, a Node.js server for API handling, a Python server for specialized services, and a MySQL database for data storage.

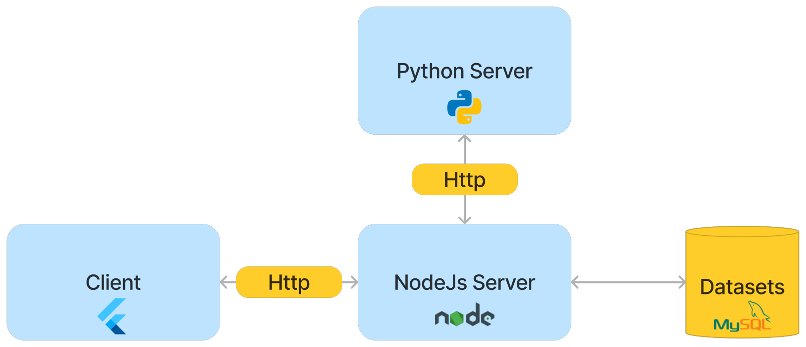


Fig. 5.1: System Architecture

This architecture efficiently distributes tasks among specialized components, ensuring scalability, maintainability, and enhanced performance. Each component will be explained in detail in the next section.

2.2 Components and their interactions

Client (Flutter)

- The client side of the application is built using Flutter, which is a framework for building cross-platform mobile, web, and desktop applications.
- It provides the user interface and interacts with the server through HTTP requests, enabling users to interact with the application.

Python Server

- Provides specialized services such as a recommendation system and a chatbot.
- Utilizes machine learning and natural language processing (NLP) to deliver personalized recommendations and intelligent chatbot responses.

MySQL Database

- Serves as the primary data storage system.
- Stores datasets that include user information, application data, and any other relevant information required by the Node.js server.

2.3 Technology stack used

- **Flutter:** Flutter is an open-source UI software development kit created by Google. It allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase using the Dart programming language.

Fig. 5.2: Flutter logo

- **Nodejs:** Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It is designed to build scalable network applications and is known for its non-blocking, event-driven architecture.

Fig. 5.3: NodeJs logo

- **MySql:** MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). It is widely used for web database applications due to its reliability, ease of use, and performance.

Fig. 5.4: MySql logo

- **Socket.io:** Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bidirectional, and event-based communication between web clients and servers, commonly used for chat applications and live updates.

Fig. 5.5: Socket.IO logo

- **Agora:** Agora is a cloud-based platform providing APIs for real-time video and audio communications. It offers tools for embedding high-quality video, voice, and live interactive broadcasting into applications across various platforms.

Fig. 5.6: Flutter logo

3 Frontend Development

3.1 Clean Architecture

Clean Architecture is a software design pattern that separates the concerns of an application into distinct layers, promoting scalability, maintainability, and testability. This architecture is particularly useful in frontend development as it ensures that the user interface remains decoupled from the business logic and data access layers. The main components of Clean Architecture include the Presentation Layer, Domain Layer, and Data Layer. Each layer has a specific role, allowing developers to manage complexity and make the codebase easier to understand and evolve.

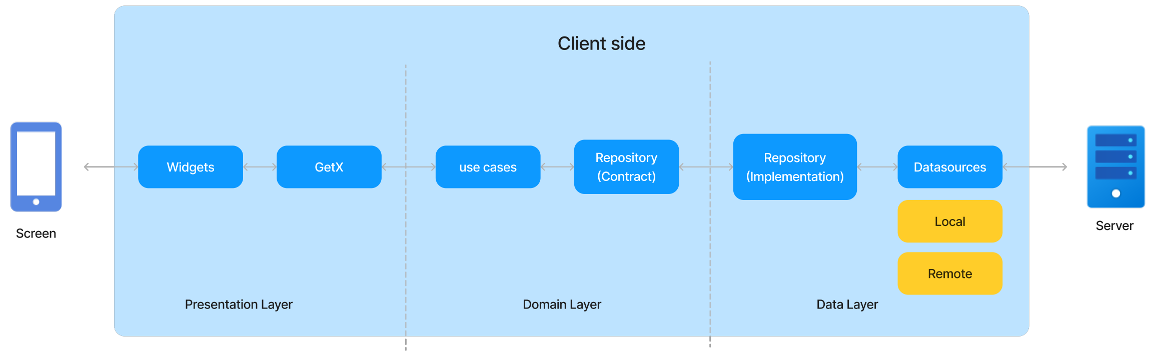


Fig. 5.7: Clean Architecture

3.1.1 Presentation Layer

The Presentation Layer is responsible for displaying information to the user and handling user interactions.

Components:

- **Widgets:** These are the building blocks of the user interface in Flutter. They define the visual elements and layout of the application.
- **GetX:** This state management solution helps manage the state of the application, making it reactive and allowing for efficient UI updates. While there are many solutions for state management in Flutter, we chose GetX due to its ease of use and simplicity, which significantly enhances developer productivity.

3.1.2 Domain Layer

The Domain Layer encapsulates the core business logic and rules of the application.

Components:

- **Use Cases:** These represent the specific actions or operations that a user can perform within the application. They define the interaction between the user interface and the business logic.
- **Repository (Contract):** This defines the interface for data operations, ensuring that the business logic does not depend on the specifics of data retrieval or storage mechanisms.

3.1.3 Data Layer

The Data Layer is responsible for data retrieval, storage, and management.

Components:

- **Repository (Implementation):** This is the concrete implementation of the repository interface defined in the Domain Layer. It contains the logic for accessing data from various sources.
- **Datasources:** These are the actual sources of data, which can be local (e.g., SQLite database on the device) or remote (e.g., REST API).
  - **Local:** Handles data storage on the client's device, enabling offline functionality.
  - **Remote:** Manages data retrieval from remote servers, ensuring that the application can fetch and update data as needed.

3.2 Frameworks and tools

- **Figma:** Figma is a web-based design tool used for interface design, prototyping, and collaboration. It allows multiple designers to work simultaneously on the same project, making it ideal for team-based design workflows and real-time collaboration.
- **Flutter widgets:** Flutter widgets are the core building blocks of Flutter's user interface. They are immutable descriptions of parts of the UI and can be combined to create complex interfaces. Flutter provides a rich set of pre-designed widgets for building responsive, high-performance applications.
- **GetX:** GetX is a powerful and lightweight state management library for Flutter. It offers a range of features including state management, dependency injection, and route management, providing an all-in-one solution to simplify and enhance Flutter application development.

3.3 User interface design

The User Interface (UI) for our platform was meticulously designed using Figma, a leading web-based design tool. Our approach to UI design was guided by several core principles to ensure an intuitive, efficient, and user-friendly experience:

- **Simplicity and Clarity:** Minimize clutter and focus on essential elements.
- **Consistency:** Maintain uniformity in colors, typography, and layout.
- **Responsiveness:** Ensure the UI adapts seamlessly to various screen sizes.
- **Visual Hierarchy:** Implement a clear hierarchy to guide users effectively.
- **Interactive and Engaging Elements:** Incorporate hover effects, transitions, and animations.

- **Feedback Mechanisms:** Include validations, error messages, and success notifications.

Below are screenshots of various sections of the platform, illustrating our design approach:

- **Authentication**

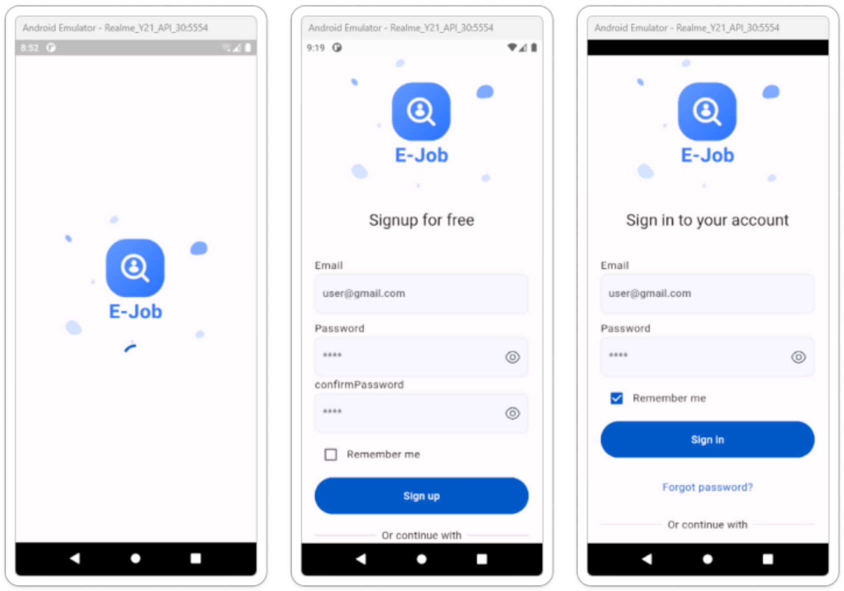


Fig. 5.8: Authentication screenshot

- **Seeker Side**

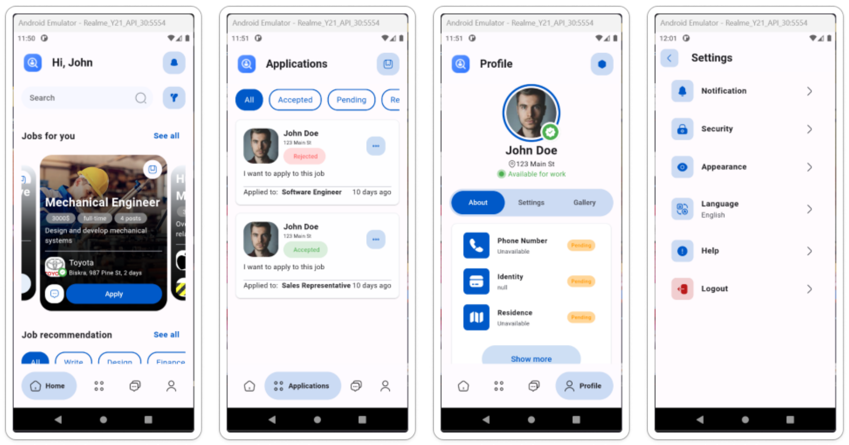
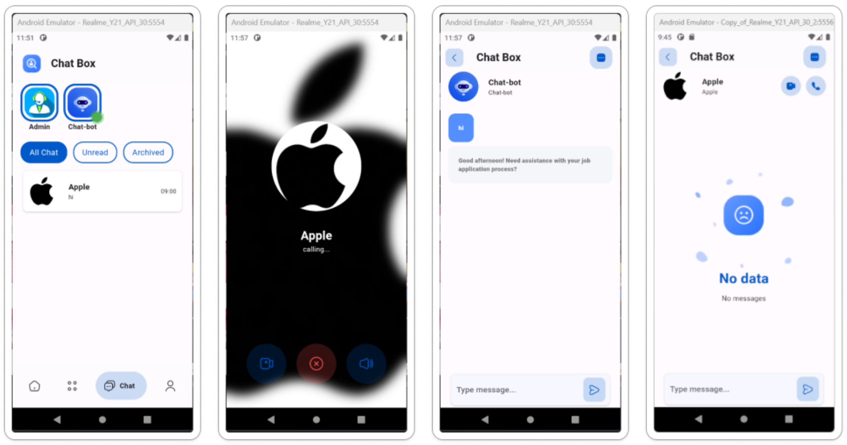


Fig. 5.9: Seeker side screenshot

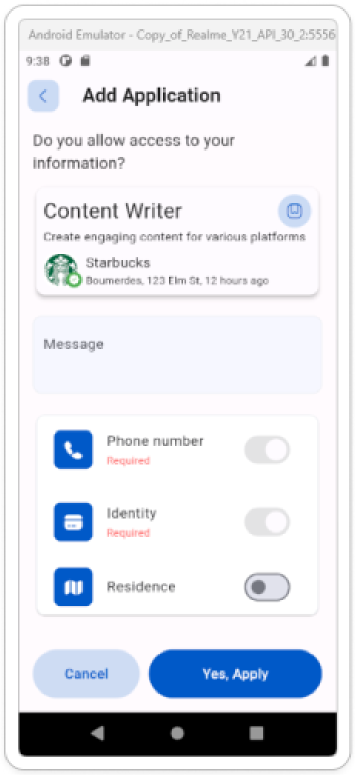
- **Messaging Pages**



Good afternoon! Need assistance with your job application process?

Fig. 5.10: Messaging pages screenshot

- **Add Application**



Android Emulator - Copy_of_Realme_Y21_API_30_2:5556

9:38

Add Application

Do you allow access to your information?

Content Writer

Create engaging content for various platforms

Starbucks

Boumerdes, 123 Elm St, 12 hours ago

Message

Phone number
Required

Identity
Required

Residence

Cancel Yes, Apply

Fig. 5.11: Add application screenshot

- **Recruiter Side**

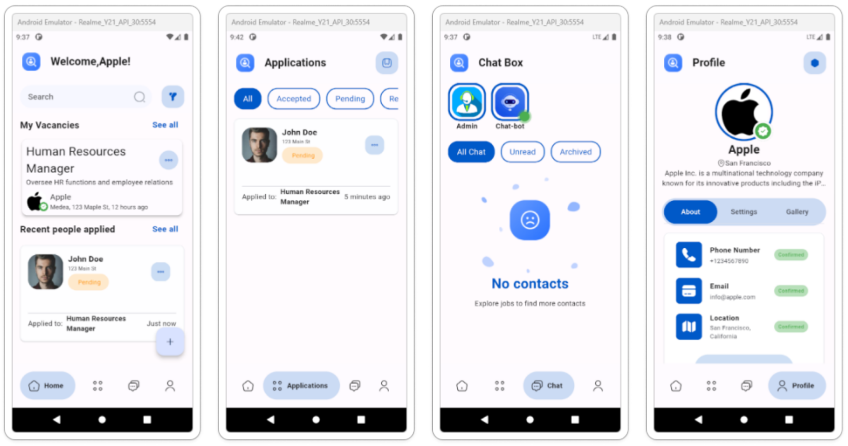


Fig. 5.12: Recruiter side screenshot

- **Publish Job**

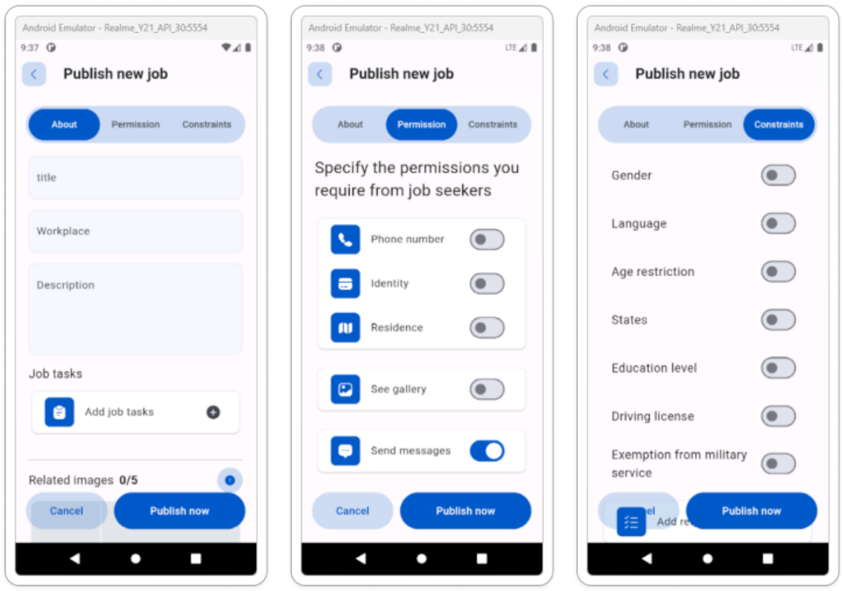
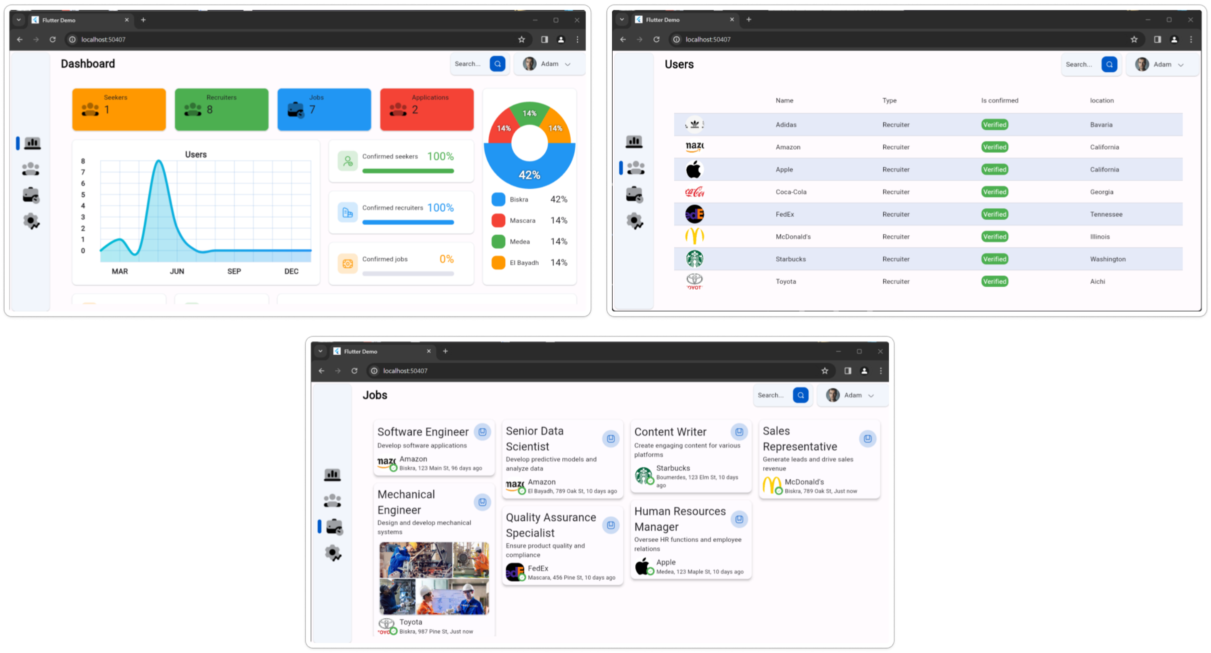


Fig. 5.13: Publish job screenshot

- **Admin dashboard**



Users

Name	Type	Is confirmed	location
Adidas	Recruiter	Verified	Bavaria
Amazon	Recruiter	Verified	California
Apple	Recruiter	Verified	California
Coca-Cola	Recruiter	Verified	Georgia
FedEx	Recruiter	Verified	Tennessee
McDonald's	Recruiter	Verified	Illinois
Starbucks	Recruiter	Verified	Washington
Toyota	Recruiter	Verified	Aichi

Jobs

Software Engineer
Develop software applications
Amazon
Biskra, 123 Main St, 96 days ago

Senior Data Scientist
Develop predictive models and analyze data
Amazon
El Bayadh, 789 Oak St, 10 days ago

Content Writer
Create engaging content for various platforms
Starbucks
Boumerdes, 123 Elm St, 10 days ago

Sales Representative
Generate leads and drive sales revenue
McDonald's
Biskra, 789 Oak St, Just now

Mechanical Engineer
Design and develop mechanical systems
Toyota
Biskra, 987 Pine St, Just now

Quality Assurance Specialist
Ensure product quality and compliance
FedEx
Mascara, 456 Pine St, 10 days ago

Human Resources Manager
Oversee HR functions and employee relations
Apple
Medea, 123 Maple St, 10 days ago

Fig. 5.14: Admin dashboard screenshot

4 Backend Development

4.1 MVC Architecture

The Model-View-Controller (MVC) architecture is a widely used design pattern for developing APIs. It separates the application into three interconnected components, each with a distinct responsibility, enhancing modularity and allowing for more maintainable and scalable API code.

We chose the MVC architecture for its simplicity and ease of use, allowing developers to create well-structured applications without requiring expert-level knowledge. Its clear separation of concerns makes it accessible for developers at all skill levels to build and maintain web applications efficiently.

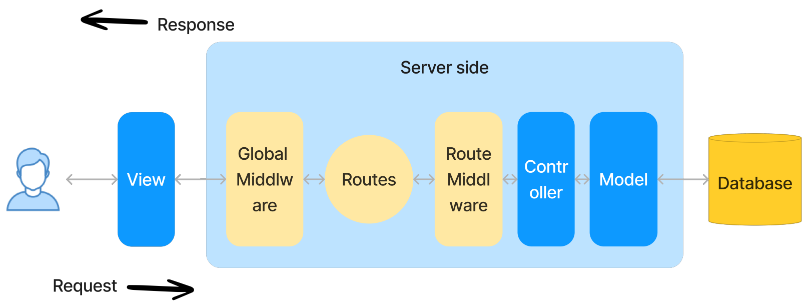


Fig. 5.15: MVC Architecture

4.1.1 Model

- The Model represents the data and the business logic of the application. It directly manages the data, logic, and rules of the application.
- It interacts with the database, retrieves data, performs operations on it, and sends the processed data back to the controller.

4.1.2 View

- The View is responsible for the presentation layer of the application. It displays the data provided by the Controller in a format suitable for interaction.
- It receives user input and requests, which it forwards to the controller for processing. It then receives data from the controller to render and display to the user.

4.1.3 Controller

- The Controller acts as an intermediary between the Model and the View. It handles incoming requests, processes them (often involving calls to the Model), and determines the appropriate response.
- It receives input from the View, processes the data with the help of the Model, and returns the processed data to the View for presentation.

4.1.4 Workflow in MVC Architecture

1. **User Interaction:** The process begins when the user sends a request from the client side, which is directed to the server.
2. **Request Handling:** The server receives incoming requests and forwards them to the appropriate Global Middleware.
3. **Global Middleware:** This layer can handle cross-cutting concerns such as authentication, logging, or data parsing before the request reaches the Routes.
4. **Routing:** The Routes determine which controller should handle the incoming request based on the request's URL and method.
5. **Route Middleware:** This middleware can handle route-specific concerns, such as permissions or validation, before the request reaches the Controller.
6. **Controller Logic:** The Controller processes the request. If data manipulation or retrieval is needed, the Controller interacts with the Model.
7. **Model Interaction:** The Model performs the necessary operations on the database and returns the results to the Controller.
8. **Response Assembly:** The Controller then processes the data and sends it back to the View.

9. **View Rendering:** The View generates the final output (e.g., HTML, JSON) based on the data from the Controller and sends it back to the client as a response.

4.2 Server-side Frameworks and tools

- **ExpressJs:** Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It facilitates the development of server-side applications by simplifying routing, middleware management, and HTTP handling.

- **Prisma:** Prisma is an open-source database toolkit and ORM (Object-Relational Mapping) for Node.js and TypeScript. It simplifies database management by providing a type-safe query builder and an intuitive data modeling tool, enhancing developer productivity and ensuring type safety.

- **EncryptJs:** EncryptJS is a JavaScript library that provides simple and effective methods for data encryption and decryption. It enables developers to secure sensitive information by using algorithms such as AES (Advanced Encryption Standard) to protect data within their applications.

- **Thunder Client:** Thunder Client is a lightweight, fast, and intuitive REST API client extension for Visual Studio Code. It allows developers to test and debug APIs directly within the code editor, streamlining the development workflow and improving productivity.

4.3 API

A RESTful API (Representational State Transfer) is an architectural style for designing networked applications. It uses standard HTTP methods like GET, POST, PUT, and DELETE to perform CRUD operations on resources, which are typically represented in JSON format. This approach enables seamless communication between clients and servers, promoting scalability and simplicity in web services.

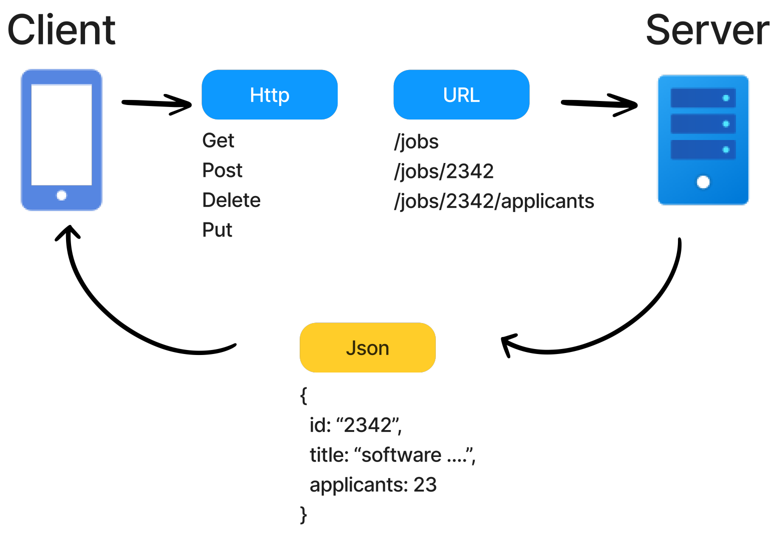


Fig. 5.16: RESTful API

We used RESTful API in our system because it provides a flexible, scalable, and easy-to-maintain way to handle client-server communication. Its stateless nature ensures that each request contains all the information needed for the server to process it, enhancing efficiency and reliability.

4.3.1 HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is the foundation of any data exchange on the Web and a protocol used for transmitting hypermedia documents, such as HTML. It is designed for communication between web browsers and web servers, but it can also be used for other purposes.

Methods

- **GET:** Retrieve data from the server (e.g., fetching a web page).
- **POST:** Submit data to the server (e.g., submitting a form).
- **PUT:** Update existing data on the server.
- **DELETE:** Remove data from the server. : Apply partial modifications to a resource.
- **HEAD:** Retrieve the headers of a resource without fetching its body.
- **OPTIONS:** Describe the communication options for the target resource.

Headers

HTTP headers are used to pass additional information with the request or response. Common headers include Content-Type (indicates the media type of the resource), Authorization (contains credentials to authenticate a user), and User-Agent (identifies the client software).

Status Codes

HTTP responses include status codes that indicate the outcome of the request. Common status codes include:

- **200 OK:** The request was successful.
- **201 Created:** The request was successful and a new resource was created.
- **400 Bad Request:** The request was malformed or invalid.
- **401 Unauthorized:** Authentication is required and has failed or has not yet been provided.
- **403 Forbidden:** The server understands the request but refuses to authorize it.
- **404 Not Found:** The requested resource could not be found.
- **500 Internal Server Error:** The server encountered an unexpected condition that prevented it from fulfilling the request.

4.3.2 JSON

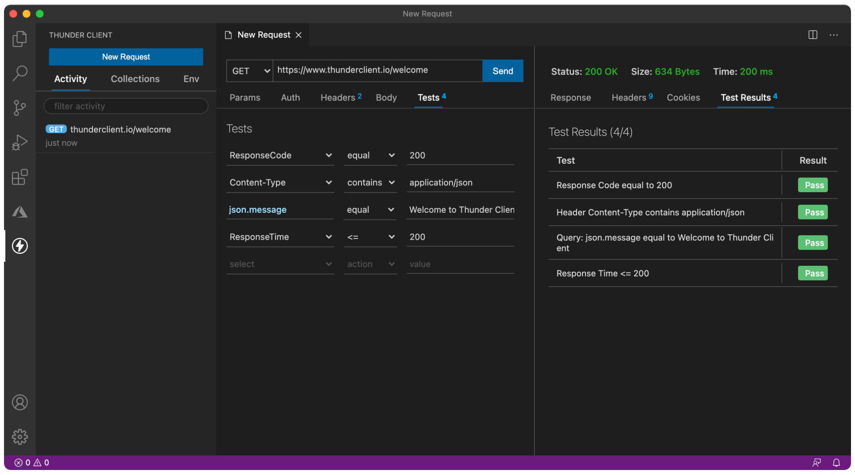
JSON, or JavaScript Object Notation, is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is commonly used to transmit data between a server and a web application as an alternative to XML.

We chose JSON for our system for several compelling reasons:

1. **Simplicity:** JSON's straightforward syntax, which consists of key-value pairs and arrays, makes it easy to learn, read, and write.
2. **Language Independence:** JSON is language-agnostic, meaning it can be used and parsed by virtually any programming language. This makes it ideal for data interchange between diverse systems and platforms.
3. **Efficiency:** JSON's lightweight nature ensures that data is transmitted efficiently over the network, reducing bandwidth usage and improving performance.
4. **Human-Readable:** JSON's structure is easy for humans to read and understand, which simplifies debugging and development processes.
5. **Wide Adoption:** JSON is widely supported across different web technologies, frameworks, and libraries. This extensive adoption ensures compatibility and ease of integration with various tools and services.

4.3.3 API Testing

To ensure the reliability and correctness of our RESTful APIs, we used Thunder Client for testing. Thunder Client is a lightweight and easy-to-use extension for Visual Studio Code, designed for testing APIs. It allows us to quickly create and execute API requests, check responses, and validate the behavior of our endpoints. This tool helps us identify and fix issues early in the development process, ensuring that our APIs function as expected and meet the required specifications.



GET https://www.thunderclient.io/welcome

Status: 200 OK Size: 634 Bytes Time: 200 ms

Tests

ResponseCode	equal	200
Content-Type	contains	application/json
json.message	equal	Welcome to Thunder Clien
ResponseTime	<=	200
select	action	value

Test Results (4/4)

Test	Result
Response Code equal to 200	Pass
Header Content-Type contains application/json	Pass
Query: json.message equal to Welcome to Thunder Client	Pass
Response Time <= 200	Pass

Fig. 5.17: Thunder client screenshot

4.4 Database design

The database design for our platform is structured to efficiently handle and organize data related to job postings, users, applications, and the interactions between them. The schema includes the following primary entities: JobApplication, Job, User, Recruiter, Seeker, and SeekerSkill. Additionally, there are other secondary entities, such as Message, Contact, and SavedJob, which support supplementary functionalities but are not crucial to include in this overview.

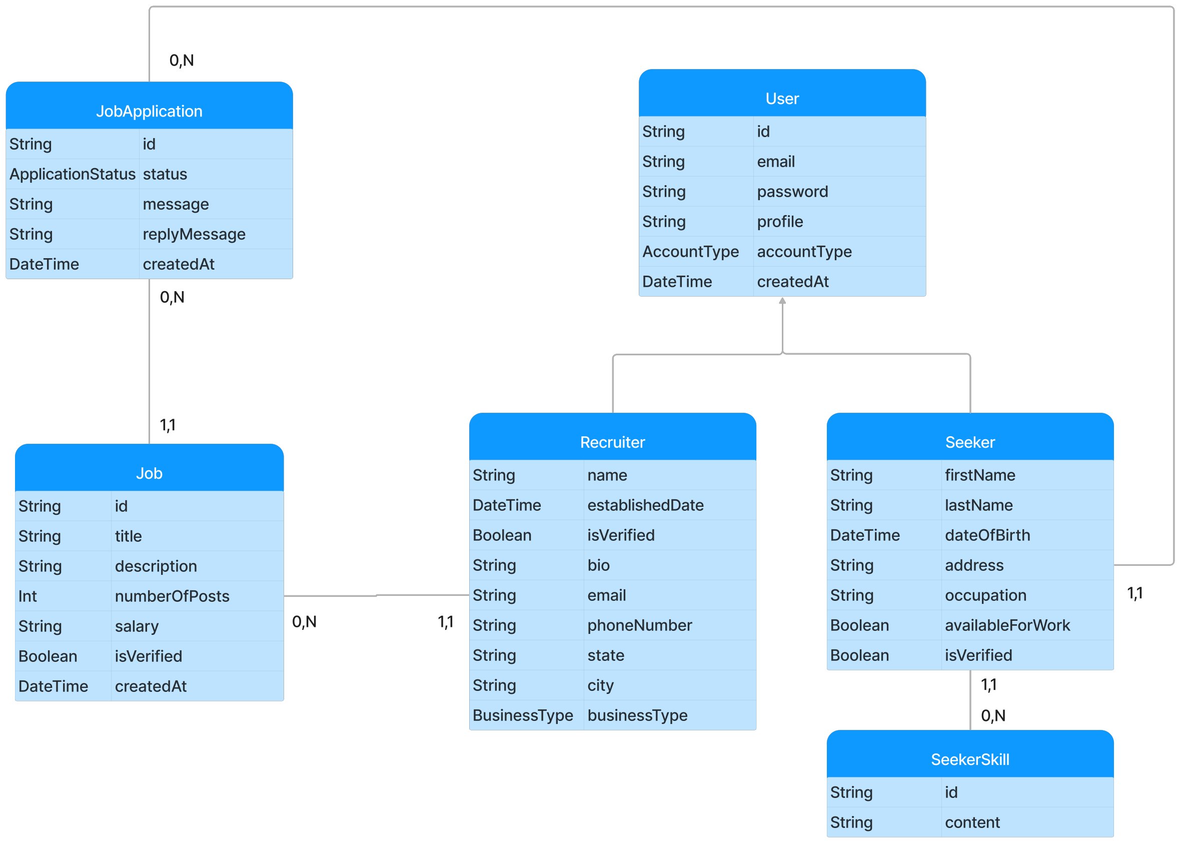


Fig. 5.18: Entity-Relationship Diagram for database

4.5 Security considerations

Ensuring the security of our application is crucial for protecting user data and maintaining the integrity of the system. This section discusses various security measures, including the use of JSON Web Tokens (JWT) for authentication, and other important aspects of application security.

JSON Web Tokens (JWT) Authentication

JWT is a widely used method for securely transmitting information between parties. Here's how JWT can be used to enhance the security of our application:

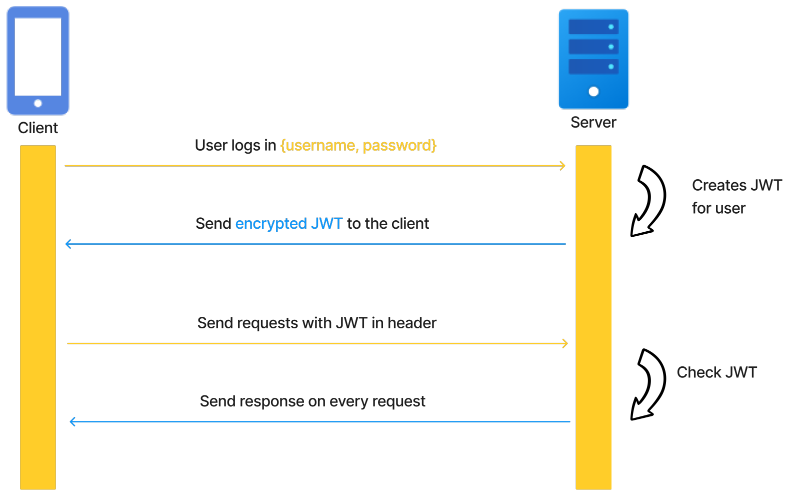

Client
Server
User logs in {username, password}
Creates JWT for user
Send encrypted JWT to the client
Send requests with JWT in header
Check JWT
Send response on every request

Fig. 5.19: JSON Web Tokens

1. **User Authentication:** Users log in with their username and password. These credentials are sent to the server for verification.

2. **JWT Creation and Encryption:** Upon successful authentication, the server creates a JWT containing the user's identity and other claims. This token is encrypted to ensure its integrity and security.

3. **Token Transmission:** The encrypted JWT is sent back to the client. The client stores this token securely, typically in local storage or a secure cookie.

4. **Token-Based Requests:** For all subsequent requests, the client includes the JWT in the HTTP headers. This allows the server to authenticate the user without requiring them to re-enter their credentials.

5. **Token Verification:** The server verifies the JWT on each request. It checks the token's signature and expiration date to ensure its validity. If the token is valid, the server processes the request and sends a response.

In addition to JWT, several other security practices are essential for maintaining the overall security of your application:

- **Data Encryption:** Use HTTPS to encrypt data transmitted between the client and server. This prevents interception by malicious actors.

- **Input Validation and Sanitization:** Validate and sanitize all user inputs to protect against injection attacks, such as SQL injection and cross-site scripting (XSS).

- **Rate Limiting and Throttling:** Implement rate limiting to control the number of requests a user can make within a specified time period. This helps prevent brute-force attacks and reduces server load.

- **User Session Management:** Manage user sessions effectively, including implementing session expiration and revocation mechanisms to prevent unauthorized access.

- **Secure Storage of Sensitive Data:** Store sensitive data, such as user credentials and tokens, securely. Use strong hashing algorithms for passwords and ensure tokens are stored in secure, non-accessible client-side storage.

- **Access Control:** Implement robust access control mechanisms to ensure that only authorized users can access certain parts of the application. This includes validating user roles and permissions.

5 Integration of the Recommendation System

The integration of the recommendation system into the platform is a critical component that enhances the platform's ability to provide personalized job suggestions to users. The architecture of this integration is depicted in Figure 5.20.

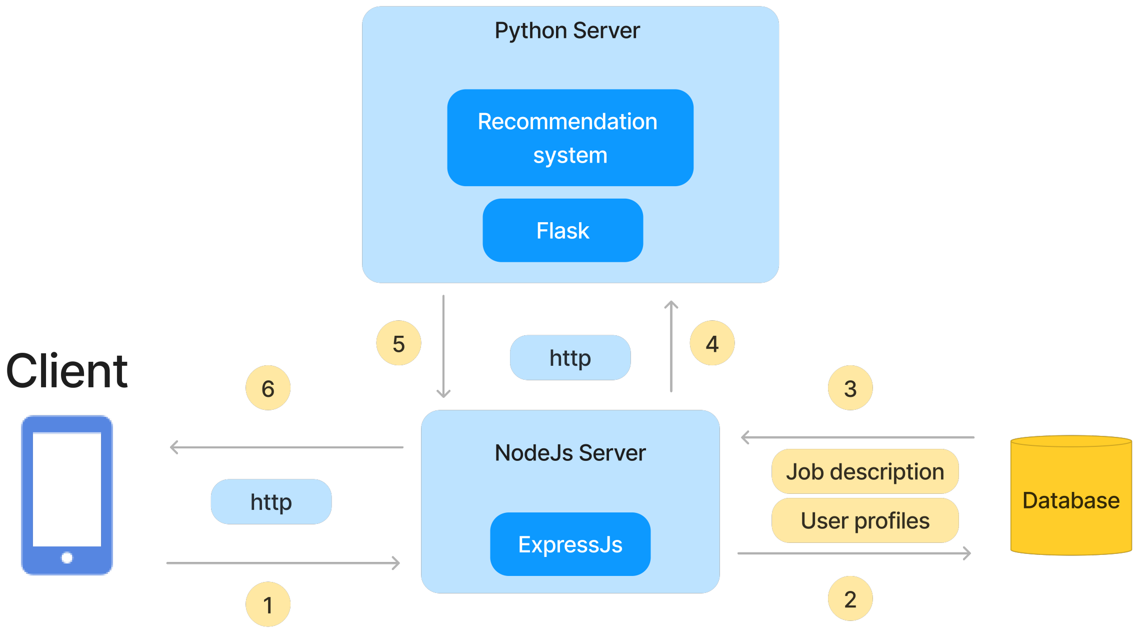


Fig. 5.20: Recommendation System Integration

The process is organized into a sequence of interactions between the client, the Node.js server, the Python server running the recommendation system, and the database.

1. **Client Request:** The process begins when the client (user) makes a request through the mobile application or web interface. This request is sent to the Node.js server via HTTP.

2. **Node.js Server:** The Node.js server, built with Express.js, acts as the intermediary between the client and the backend systems. It receives the client's request and processes it, ensuring it is in the correct format for further processing.

3. **Database Interaction:** The Node.js server then interacts with the database to fetch relevant data such as job descriptions and user profiles. This data is essential for generating accurate recommendations and is retrieved from various tables within the database.

4. **Data Transfer to Python Server:** Once the necessary data is gathered, it is sent to the Python server, which hosts the recommendation system. The communication between the Node.js server and the Python server is facilitated through HTTP requests.

5. **Recommendation System Processing:** The Python server, running a Flask application, processes the data using advanced AI algorithms. The recommendation system utilizes techniques such as TF-IDF, Bag of Words, and Word Embeddings to extract features from job descriptions and user profiles. It then calculates cosine similarity to generate job recommendations tailored to the user's preferences and history.

6. **Response Back to Node.js Server:** After processing the data, the recommendation system sends the results back to the Node.js server. This includes the list of recommended jobs for the user.

7. **Client Response:** Finally, the Node.js server sends the recommendations back to the client. The client application displays these job recommendations to the user, enhancing their job search experience with personalized suggestions.

6 Additional Features

6.1 Chat-Bot

A chatbot is an AI-driven program designed to simulate human conversation through text or voice interactions. It can assist with various tasks, from answering questions to providing customer support, by understanding and responding to user inputs.

The Chat-Bot feature is an essential component of our platform, designed to enhance user interaction and provide instant assistance. Here's a detailed explanation of how our Chat-Bot works:

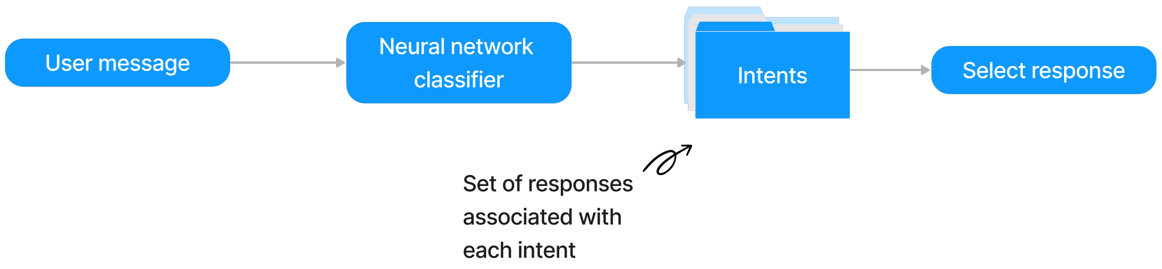
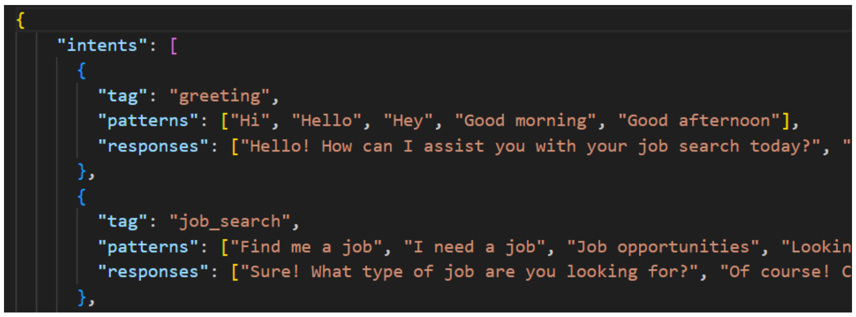

User message
Neural network classifier
Intents
Select response
Set of responses associated with each intent

Fig. 5.21: JSON Web Tokens

1. **User Message Handling:** The Chat-Bot interaction begins when a user sends a message. This message could be any query or request for assistance.
2. **Text Preprocessing:** The received message undergoes text preprocessing. This step involves cleaning the text by removing any unnecessary characters, normalizing the text (e.g., converting it to lowercase), and possibly stemming or lemmatizing words to their root forms.
3. **Feature Extraction:** After preprocessing, the message is converted into a set of features using the Bag of Words model. This involves creating a vector that represents the frequency or presence of words in the message relative to a larger corpus.
4. **Intent Classification:** The feature vector is then fed into a neural network classifier. This classifier, trained on a large dataset of categorized messages, analyzes the vector to determine the intent behind the user's message. The structure of intents is illustrated in the following figure.



```
{
    "intents": [
      {
        "tag": "greeting",
        "patterns": ["Hi", "Hello", "Hey", "Good morning", "Good afternoon"],
        "responses": ["Hello! How can I assist you with your job search today?", "
      },
      {
        "tag": "job_search",
        "patterns": ["Find me a job", "I need a job", "Job opportunities", "Lookin
        "responses": ["Sure! What type of job are you looking for?", "Of course! C
      },
```

Fig. 5.22: Intent structure

5. **Response Selection:** Once the intent is identified, the Chat-Bot selects an appropriate response from a predefined set of responses associated with that intent. The selection process may involve randomly choosing a response from the set to ensure variability in interactions.

By following these steps, our Chat-Bot efficiently processes user messages and provides relevant and dynamic responses, significantly enhancing the user experience on our platform.

6.2 Voice & Video Calling

Video Calling enables one-to-one or small-group video interactions with smooth, jitter-free streaming video. Agora's Video SDK makes it easy to embed real-time video chat into web, mobile, and native apps. With Agora's advanced network technology, we can ensure the highest available video and audio quality.

This section explains how we can integrate Video Calling features into our app. The following figure shows the basic workflow needed to integrate this feature into our app.

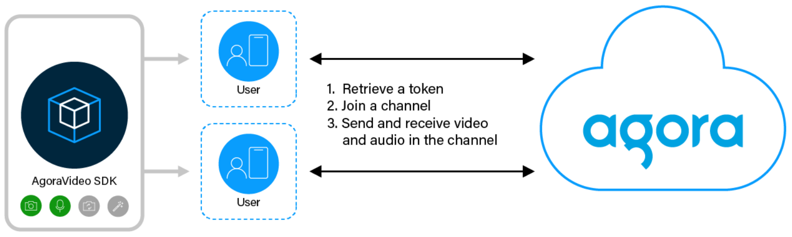


Fig. 5.23: JSON Web Tokens

To start a session, we implement the following steps in our app:

1. **Retrieve a token:** A token is a computer-generated string that authenticates a user when our app joins a channel. In this guide, we retrieve our token from Agora Console. To see how to create an authentication server for development purposes, see Implement the authentication workflow. To develop our own token generator and integrate it into our production IAM system, read Token generators.

2. **Join a channel:** Call methods to create and join a channel; apps that pass the same channel name join the same channel.

3. **Send and receive video and audio in the channel:** All users send and receive video and audio streams from all users in the channel.

Below are the prerequisites required to set up and develop a Flutter application integrated with Agora's Video SDK.

Requirement	**Details**
Flutter Version	Flutter 2.0.0 or higher
Dart Version	Dart 2.15.1 or higher
IDE Support	Android Studio, IntelliJ, VS Code, or any other IDE that supports Flutter, see Set up an editor.
Android Development	Android Studio on macOS or Windows (latest version recommended)

Continued on next page

Tab. 5.1 – continued from previous page

Agora Account and Project	An Agora account and project to generate access tokens
Internet Access	A computer with Internet access to connect with the Agora server

Tab. 5.1: Agora requirements

6.3 Real time messaging

Real-time messaging is a critical feature in modern applications, enabling instant communication between users. This section explains how to integrate real-time messaging in an application, leveraging Socket.IO for efficient and effective message delivery. The process is illustrated in the following figure.

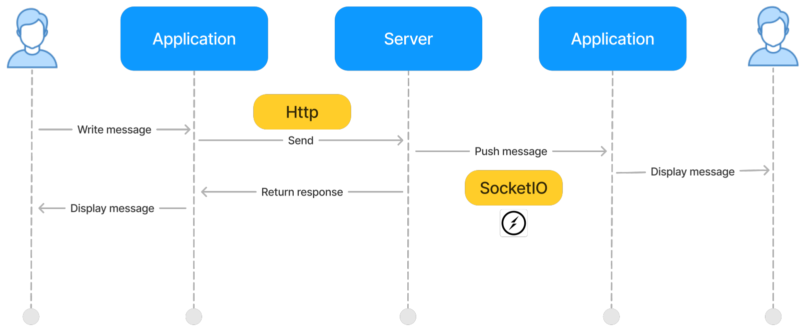


Fig. 5.24: Real time messaging

The integration of real-time messaging involves several key steps to ensure smooth communication between users. Below is a detailed breakdown of each step in the process:

1. **Write Message (Client Application - Sender):** The user initiates the messaging process by writing a message within the client application. This message is then sent to the server.

2. **Send Message (Client Application to Server):** The client application sends the message to the server. This is typically done through an HTTP request or WebSocket connection established using Socket.IO.

3. **Server Processing:** Upon receiving the message, the server processes it. This includes validating the message content, checking user permissions, and possibly logging the message for record-keeping.

4. **Return Response (Server to Client Application - Sender):** After processing the message, the server returns a response to the sender's application. This response might include a confirmation that the message was received and processed successfully.

5. **Push Message (Server to Client Application - Receiver):** Using Socket.IO, the server pushes the message to the recipient's client application in real-time. Socket.IO maintains an open WebSocket connection, enabling the server to send updates instantly without the recipient needing to request new data constantly.

6. **Display Message (Client Application - Receiver):** The recipient's client application receives the pushed message and displays it to the user. This immediate delivery and display create a seamless, real-time communication experience.

7 Conclusion

In this chapter, we explored the development of an AI-powered job recommendation platform, detailing its system architecture with Flutter, Node.js, Python, and MySQL components. We discussed the integration of advanced features like chatbots and real-time communication using Agora. Security measures, including JWT authentication and data encryption, were emphasized. We also highlighted the integration of additional features for enhanced user interaction.

CHAPTER 6

General conclusion

This dissertation presents a comprehensive approach to developing an AI-powered job recommendation platform aimed at transforming the job search and recruitment landscape in Algeria. The primary objective was to leverage artificial intelligence to enhance job matching accuracy, reduce job search time, and increase employment rates, particularly among youth and recent graduates.

Throughout the study, we identified significant inefficiencies in the existing job market, such as high unemployment rates and a lack of personalized job recommendations. By integrating advanced AI algorithms capable of learning from user interactions and preferences, our platform demonstrated the potential to provide highly accurate and tailored job recommendations, thus addressing these inefficiencies.

In conclusion, this work not only highlights the effectiveness of AI in improving job matching processes but also opens avenues for future research. Future efforts could focus on incorporating additional datasets, experimenting with other deep learning techniques, and implementing more sophisticated preprocessing methods to enhance the platform's performance. Ultimately, the successful deployment of this platform has the potential to contribute significantly to reducing unemployment and underemployment in Algeria, fostering economic growth, and supporting a dynamic and skilled workforce.

References

[1] Nasreen Azad. Professionals' perspective on using ai tools in recruitment process. *Journal of AI Recruitment*, page 6, 2024.

[2] Derek Chapman and Jane Webster. The use of technologies in the recruiting, screening, and selection processes for job candidates. *International Journal of Selection and Assessment*, 11, 09 2003.

[3] Varun Chand, Kurakula Kumar, Syarul Azlina S., Seema Sabharwal, and Sivaprakasam Kumar. A study on the impact of artificial intelligence on talent sourcing. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13:1, 03 2024.

[4] What is a recommendation system? | data science. https://www.nvidia.com/en-us/glossary/recommendation-system/, 2024. Accessed: April 2024.

[5] Akshay Kulkarni et al. *Applied Recommender Systems with Python: Build Recommender Systems with Deep Learning, NLP and Graph-Based Techniques*. Apress, 2022. Accessed: 2024-05-05.

[6] Zhenhua Dong et al. A brief history of recommender systems. *Journal of Recommendation Engines*, 2022.

[7] Nunung Nurul Qomariyah. Definition and history of recommender systems. https://international.binus.ac.id/computer-science/2020/11/03/definition-and-history-of-recommender-systems/, 2020. Accessed: April 2024.

[8] Corinna Underwood. Use cases of recommendation systems in business – current applications and methods. https://emerj.com/ai-sector-overviews/use-cases-recommendation-systems/, March 2020. Accessed: April 2024.

[9] IT Convergence. Top use cases of ai-based recommendation systems. https://www.itconvergence.com/blog/top-use-cases-of-ai-based-recommendation-systems/, 2023. Accessed: 2024-05-05.

[10] Carole Sonia et al. Enhanced experiences: Benefits of ai-powered recommendation systems. *Journal of AI Technologies*, pages 216–220, 2024.

[11] Zeshan Fayyaz et al. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences*, 20(21), 2020.

[12] Frank Edwin et al. Overview of natural language processing (nlp) in bi. *Journal of Business Intelligence*, pages 5–11, 2024.

[13] Masato Hagiwara. *Real-World Natural Language Processing: Practical applications with deep learning*. Manning, 3 edition, 2021.

[14] Elizabeth D. Liddy. *Natural Language Processing*, page 15. 2001.

[15] Akshay Kulkarni and Adarsha Shivananda. *Natural Language Processing Recipes: Unlocking Text Data with Machine Learning and Deep Learning Using Python*. Apress, 2 edition, 2021.

[16] Rajvardhan Patil et al. A survey of text representation and embedding techniques in nlp. *IEEE Access*, PP, 2023.

[17] Schutte Dalton and Zhang Rui. Considerations for specialized health ai & ml modelling and applications: Nlp. *Healthcare AI*, pages 623–641, 2024.

[18] M. Deza and Elena Deza. *Encyclopedia of Distances*. Springer, 2009.

[19] KANDI JAGADISH. Job analysis content recommendation, 2019. Accessed: May 2024.