A Work Project, presented as part of the requirements for the Award of a Master's degree in Business Analytics from the Nova School of Business and Economics.

Skillmatch.ai: Applying a Skills First Framework to Matching Algorithms –

Proof of Concept for Evaluating and Optimizing the Model Pipeline Using Synthetic Data

Luca Oeztekin (59168)

Work project carried out under the supervision of:

Michael Kummer

17-12-2024

Abstract:

The rapid evolution of technology and automation is reshaping the global workforce, increasing the demand for new skills. This paper investigates the drawbacks and potentials of replacing traditional hiring methods by a "skills-first" strategy. The dissertation will explore how prioritizing measurable competencies over formal degrees or job titles, can address pressing talent shortages and enhance workforce adaptability. Three dimensions are investigated: skill overlap among roles, the design of a skills-based ecosystem, and strategies to address data accessibility challenges in evaluating candidate-job matching models. Despite the limitations, this work advances skills-first hiring, fostering more inclusive and effective approach to talent management.

Individual Abstract:

Proof of concept for evaluating and optimizing the Skillmatch.ai model pipeline using synthetic data. It outlines the iterative benchmarking process designed to refine key pipeline components, focusing on skills representation, similarity thresholds, embedding models, and encoder architectures. Employing rigorous metrics such as F1-score, mean similarity score difference, and processing time, the evaluation highlights trade-offs between precision and operational efficiency. Insights gained inform the design of an advanced model pipeline, enabling precise matching while maintaining operational efficiency. This approach establishes a robust framework for model refinement, ensuring adaptability to real-world recruitment challenges and evolving business needs.

Keywords:

Data Science, Artificial Intelligence (AI), Machine Learning (ML), Natural Language Processing (NLP),, Large Language Models, Generative AI (Gen AI), OpenAI API, Embedding Models, Pretrained Sentence Transformer, Bi-Encoder, Cross-Encoder, Semantic Similarity, Recruiting and Talent Management, Career Development, Role Transitioning, Skills Mapping, Skill Overlap, Occupational Role Recommendations, Variance in Occupational Similarity, Soft Skills, Hard Skills, Similarity Metrics, Network Analysis, Model Performance Evaluation, Proof of Concept, Binary Classification, Synthetic Data Generation.

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

1. Introduction *4*
1.1 Research Motivation 4
1.2 Research Gap 5
1.3 Research Question 7
2 Methodology *8*
3 Literature Review *11*
3.1 Traditional Hiring 11
3.2 Skills-First Approach 13
3.3 AI in Human Resource Management 21
3.4 Technical Perspective on Matching Models in Recruitment 27
3.5 Summary of Literature Findings 35
4. Skill Exploratory Data Analysis *36*
4.1 Data Collection 37
4.2 Role Overlap 39
4.3 Industry Context and Its Impact on Skill Overlap 48
4.4 Role Overlaps in Terms of Demand 50
4.5 Future Improvements & Developments 53
4.6 Main Findings 54
5 Model Development: Skills-First Résumé Matching *56*
5.1 Motivation 56
5.2 Building the Matching Model 57
5.3 Limitations and Future Development 68
6. Generating Synthetic Data to Overcome Data Accessibility Challenges for Model Testing and Evaluation *71*
6.1 The Critical Role of Evaluation in Model Development 71
6.2 Problem Statement 72
6.3 Ideal Data Solution for Testing and Evaluation 73
6.4 Limitations of the Proposed Data Collection Approach 75
6.5 Synthetic Data Approach 76
6.6 Synthetic Data Generation Process 77
7. Proof of Concept for Evaluating and Optimizing the Model Pipeline Using Synthetic Data *82*
7.1 Evaluation Methodology and Objectives 82
7.2 Evaluation Process 83
7.3 Results 87
7.4 Conclusion: Advanced Model Pipeline 90

7.5 Limitations 91
7.6 Future Outlook 91
8. Skills-Based Similarity Model for Career Development 92
8.1 The Importance of Identifying Skill-Aligned Career Pathways 92
8.2 Skill-Based Role Recommendations Objective 93
8.3 Data Used 93
8.4 Methodology for Skill-Based Similarity Calculation 94
8.6 Limitations and Further Development 100
8.7 Key Insight 101
9. Enhancing Role Recommendations through Comprehensive Occupational Similarity Analysis 102
9.1 Motivation 102
9.2 Data and Feature Analysis 103
9.3 Role Similarity Calculation 105
9.4 Feature Selection 107
9.5 Benchmarking Features Against O*NET Rankings 109
9.6 Limitations 110
9.7 Graphical Representation of the Model Output 110
9.8 Final Output 111
9.9 Conclusion 112
10. Building an Ecosystem of Skills-First Applications 113
10.1 Motivation 113
10.2 Core Components 115
10.3 System Architecture 118
10.4 UI/UX Design 119
10.5 Data Handling and Security 119
10.6 Deployment 119
10.7 Limitations and Next Steps 120
10. 8 Conclusion 122
11 Limitations 123
12 Implications and Future Research 124
13. Conclusion 125
Bibliography 126
Appendix 146

1. Introduction

1.1 Research Motivation

In today's rapidly evolving economic landscape, driven by advancements in automation, artificial intelligence (AI) and evolving technology, the nature of work is transforming significantly. Since 2015, job skill requirements have changed by 25%, and this is projected to double by 2028 (LinkedIn 2023) and by 2030, around 85 million jobs globally may remain unfilled due to a shortage, potentially resulting in $8.5 trillion in unrealized annual revenues. This underscores the growing challenge of securing skilled talent, which emerges as a key obstacle to business transformation in (Di Battista, et al. 2023). These disparities between the skills demanded and those possessed necessitate innovative approaches. Among these, the "skills-first" approach which prioritizes candidates' skills and competencies over traditional hiring criteria such as degrees, job experience, or job titles. This method has the potential to democratize access to economic opportunities, enabling a broader and more diverse pool of qualified candidates (Alhadrami, et al. 2023). Specifically, this approach can expand the candidate pool by nearly tenfold on average, bridging significant talent gaps (LinkedIn 2023). For businesses, the skills-first approach offers several advantages, including enhanced talent mobility, improved retention rates, increased productivity, and greater equity and inclusion (Alhadrami, et al. 2023).

Several factors have contributed to emergence and growing emphasis on skills-based talent management. The shift to remote work during the COVID-19 pandemic required more flexible, skill-focused hiring. Additionally, online learning platforms and micro-credentials have made skill acquisition more accessible. Furthermore, advances in AI and big data have enhanced the ability to map jobs and skills, facilitating more accurate identification of skill requirements and gaps (Alhadrami, et al. 2023).

In 2022, the sitting President Joe Biden underscored the importance of skills-first hiring in his State of the Union Address, signalling a strong governmental push towards this approach (The White House 2022). Moreover, many companies across are adopting skills-first practices, recognizing the benefits of focusing on competencies rather than traditional credentials. However, the transition to a skills-first paradigm has been slow, with 88% of employers still rejecting highly skilled candidates lacking formal credentials (Fuller, et al. 2021).

A skills-first approach can create substantial benefits for individuals, businesses, society, and the economy as a whole (Alhadrami, et al. 2023). Our research aims to contribute to this transition by addressing the limitations above and inspire organizations to take action.

1.2 Research Gap

The growing prominence of the skills-first approach has been extensively discussed in recent literature, emphasizing its potential to address the challenges posed by a rapidly evolving job market. Parallel to this, advancements in AI and machine learning (ML) allowed the creation of tools for candidate-job matching leveraging semantic matching, natural language processing (NLP), and embedding-based techniques (Garg, et al. 2022) to improve recruitment efficiency and reducing human biases in hiring decisions (Budhwar, et al. 2022; Esch, Black and Ferolie 2019).

Despite these advancements, challenges hinder the widespread adoption of skills-first practices. The transition to skills-focused methodologies remains slow due to organizational resistance, the lack of standardized frameworks, and difficulties in skill validation (Alhadrami, et al. 2023; Fuller, Raman and Sage-Gavin 2021). Additionally, AI and ML tools face implementation challenges due to unclear best practices (Singla 2023). These gaps provide a foundation for identifying specific areas where additional research and practical solutions are required.

Research Gap 1 – Practical Methodologies for Implementing Skills-First Practices: While literature supports skills-first hiring, it lacks practical methods for real-world application. Theoretical models emphasize its importance, but actionable strategies for implementation are limited (Alhadrami, et al. 2023; Bankert, et al. 2024). Addressing this gap requires the development of technology-enabled tools and frameworks that translate skills-first principles into tangible, scalable solutions for businesses (ibid.)

Research Gap 2 – Evaluation Frameworks and Data Accessibility: The effectiveness of skills-first models depends on robust evaluation systems, but data scarcity poses challenges. Current studies lack high-quality, real-world datasets and benchmarks for evaluating candidate-job matching algorithms (Skondras, Zervas and Tzimas 2023). While synthetic data generation offers a partial solution, it remains in early stages and requires refinement to ensure it reflects real-world hiring conditions (Magron, et al. 2024; ibid.). There is a need for research that advances the development of reliable synthetic datasets and establish robust evaluation frameworks to support the adoption of skills-first methodologies.

Research Gap 3 – Post-Hire Role Matching and Career Development: Existing research primarily focuses on *pre-hire recruitment* and selection, neglecting post-hire role matching and career development. Effective internal mobility and workforce planning are critical for maximizing the potential of employees, yet tools for these purposes remain limited (De Vos, et al. 2024). Expanding the scope of research to include post-hire applications is essential to create a holistic skills-first ecosystem.

Addressing the Research Gaps: This research aims to narrow these gaps by introducing a comprehensive skills-first ecosystem that includes methodologies for embedding skills-first practices into business operations, such as tailored interview questions and customized communication templates. It also incorporates advanced skills-based matching algorithms designed for both external recruitment and internal talent development.

To address data accessibility challenges, this study leverages synthetic datasets to test and evaluate the proposed skills-first matching model under realistic conditions (Magron, et al. 2024; Skondras, Zervas and Tzimas 2023). While promising, this approach serves as a proof of concept that requires validation with actual data. Additionally, it explores post-hire applications by developing tools that support career development and internal mobility, to provide actionable insights for both individuals and organizations. By addressing these gaps, this research contributes to advancing the adoption of skills-first hiring practices and workforce planning strategies, promoting a more inclusive and effective approach to talent management.

1.3 Research Question

Research Question 1 (RQ1): *What is the extent of skill overlap among roles, and how does this overlap facilitate role transitions and support a skills-first approach in recruitment and workforce development?*

This part examines the extent of skill overlap among technology roles to determine the feasibility of role transitions. If substantial overlap exists, this supports the adoption of a skills-first approach, which can lead to improved career navigation, optimized resource allocation, and greater workforce adaptability. Identifying these overlaps also allows for targeted upskilling, to bridge skill gaps and enable smooth transitions between roles.

Research Question 2 (RQ2): *How can a skills-based ecosystem be designed to foster skills-first adoption?*

To answer this question, we will analyse how to create a comprehensive skills-first ecosystem. This involves implementing features that simplify the employer's adoption of the skills-first approach, such as a ranking system for evaluating candidates by skills, tailored interview questions to assess abilities, and personalized rejection emails with clear feedback to highlight skill gaps. Beyond the hiring process, the ecosystem also provides role recommendations to

help identify optimal career paths and to allow companies to optimize resource allocation and meet training needs.

Research Question 3 (RQ3): *How to overcome data accessibility challenges for required model optimization and evaluation of skills-first algorithms?*

We will explore the use of synthetic data to ensure that skills-first algorithms can be evaluated and optimized. By generating realistic datasets, we can bypass data accessibility challenges and test a variety of matching models, providing a realistic first proof of concept.

2 Methodology

This project adopts the CRISP-DM (Cross Industry Standard Process for Data Mining) framework (Wirth and Hipp 2000), recognized as the de facto standard for data mining projects. CRISP-DM is industry-independent and ensures a structured, iterative approach to develop data-centric models (Schröer, Kruse and Gómez 2021). Its flexibility and alignment with business objectives make it well-suited for handling skills data in workforce development. The following sections describe how the six iterative phases of CRISP-DM guided this project.

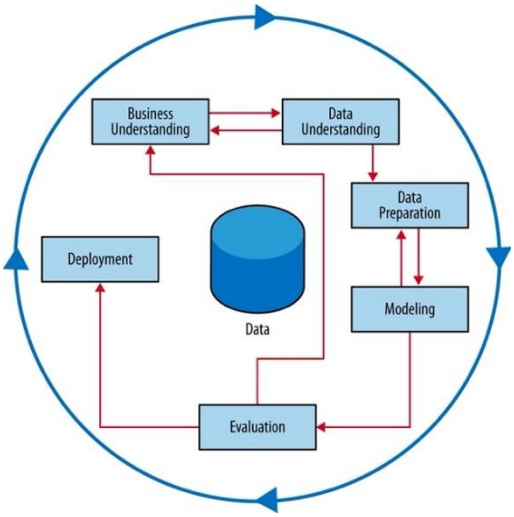


Figure 1: CRISP-DM - Wirth, R. & Hipp, J. (2000): *This figure illustrates the six iterative phases of the CRISP-DM framework. It enables iterative refinement and alignment with business objectives, making it ideal for structuring the design of ecosystem models working with highly complex skills data.*

The first phase, **Business Understanding**, forms the foundation of the project. This phase is executed in the literature review, where we explored talent shortages as a barrier to industry transformation across sectors. Through this process, we identified the skills-first approach as a promising solution, particularly in the digital and tech sectors, and aligned it with the broader business problem of improving workforce adaptability and talent acquisition (Chapter 3).

The **Data Understanding** and **Data Preparation phases** explore skills data and frame the first research hypotheses. These phases assess skill overlap in tech-related roles, supporting the feasibility of role transitions and the skills-first approach. While the exploratory data is not used

in model development, it provides valuable insights supporting the hypothesis, and ensures the research remains solution-oriented and aligned with the project's objectives (Chapter 4).

In the **Modelling** phase, we developed a candidate-job matching model that exclusively matches candidates and roles based on their skills. This model demonstrates how the skills-first approach can be implemented through algorithms, ensuring that the methodology supports competency-based recruitment practices (Chapter 5).

In the **Evaluation and Testing** phase involved assessing the model pipeline. Due to GDPR restrictions on accessing real résumé data, we utilized synthetic datasets to evaluate the model's performance. Despite the limitations of synthetic data, this phase ensured the model was iteratively improved to align with the project's business objectives (Chapters 6 and 7).

Finally, the **Deployment** phase focuses on translating the model into a functional web application. The application facilitates job-résumé matching and supports additional use cases. This phase ensures the practical usability of the system by recruiters and job seekers, creating a scalable and user-friendly ecosystem (Chapters 8, 9, and 10).

By following the six iterative phases of the CRISP-DM framework, this project follows a structured, comprehensive approach, ensuring alignment with business objectives and delivering an impactful ecosystem solution.

3 Literature Review

In this literature review, we examine existing research on skills-first hiring practices and explore technical solutions for candidate-job matching, which serve as the foundation for proposing our matching model pipeline. Relevant academic databases were identified, focusing on peer-reviewed journal articles, books and research papers. The review employed a keyword-based search strategy and sources were selected based on journal rankings, citation frequency, and relevance to ensure a solid foundation for this work. As the skills-first approach is an emerging topic with limited availability of recent academic papers, industry reports were also utilized to complement the literature review and provide an overview of the field.

Building on this methodology, the review explores three key areas of focus. First, it compares traditional hiring criteria with the skills-first approach, highlighting the shift toward measurable competencies and adaptability. Second, it examines the skills-first approach in greater depth, analysing its benefits, such as improved alignment and flexibility, alongside challenges like skill validation and bias. Lastly, it explores advancements in technology within human resource management, particularly the integration of ML and NLP tools to enhance efficiency in candidate-job matching.

3.1 Traditional Hiring

In this project, traditional hiring is defined as the approach where hiring managers evaluate candidates primarily based on educational background, job experience, perceived personal fit, and other conventional indicators. In contrast to the skills-first approach, traditional hiring relies on these credentials as proxies for a candidate's potential fit and accumulated human capital, rather than directly assessing the specific skills required for the position.

3.1.1 Traditional Hiring Criteria

One of the primary criteria in traditional hiring is **education titles**, which organizations often view as proxies for skills and knowledge, particularly in fields requiring specialized training (Wilk and Cappelli 2006; Gusdorf 2008). Another key criterion is **experience**, which is frequently valued in traditional hiring due to its perceived role in predicting job performance. When experience aligns with the relevant industry or job function, it is assumed to enable candidates to transfer their accumulated skills and knowledge effectively to new roles (Dokko, Wilk and Rothbard 2009; McDaniel 1988). Additionally, traditional hiring emphasizes the personal fit, or the alignment of a candidate's personality and values with the organization's culture, as an influence factor to long-term retention and performance. This concept of person-organization fit is often evaluated alongside skills and knowledge to determine whether a candidate will integrate well into the team and adapt to organizational norms (Bowen, Ledford and Nathan 1991; Kristof 1996). That means personal attributes such as adaptability and interpersonal skills are frequently screened in traditional hiring to complement qualifications, further enhancing the perceived fit with organizational goals (Bowen, Ledford and Nathan 1991). Despite the longstanding reliance on these criteria, research increasingly highlights the limitations of traditional hiring practices. These limitations underscore the need for alternative approaches, such as skills-first hiring, which may better capture the dynamic and skill-specific requirements of modern roles.

3.1.2 Limitations of Traditional Hiring Approaches

Traditional hiring approaches, while long-standing and widely used, face several limitations that can reduce the effectiveness of talent selection and workforce diversity. Selection frameworks that focus on formal education and degree titles can unintentionally exclude candidates who have developed relevant skills through alternative learning paths, such as online courses, bootcamps, or hands-on experience. This reliance on degree-based qualifications often

creates barriers for skilled candidates without formal credentials, narrowing the talent pool and reducing workforce diversity (Fuller, Langer and Sigelman 2022; Hancock, et al. 2022). This reliance on academic credentials may result in ineffective job matching, as degrees do not always reflect a candidate's practical abilities, potentially leading to lower job performance and reduced job engagement (Hancock, et al. 2022).

While experience is often used as an indicator of job readiness, it poses certain limitations as a hiring criterion. Research indicates that prior experience may not always lead to optimal performance, particularly when a role requires specific skills and knowledge. Established cognitive patterns and routines from previous positions can reduce adaptability, making it harder for candidates to adapt to different organizational contexts (Dokko, Wilk and Rothbard 2009; Beyer and Hannah 2002). Studies suggest that a more balanced approach, which values both skill relevance and adaptability alongside experience, leads to better job performance than experience alone (Beyer and Hannah 2002).

Furthermore, research highlights the limited predictive value of pre-hire work experience on long-term performance, training success, and turnover rates. While task-specific experience may enhance initial performance, its effect typically declines over time as employees acquire on-the-job skills. This suggests that organizations should be cautious in overemphasizing prior experience during selection, as it is often minimally related to sustained performance outcomes (Van Iddekinge, et al. 2019).

These findings highlight the need for organizations to move beyond traditional hiring criteria and adopt a more holistic approach that incorporates a broader range of qualifications when evaluating candidates.

3.2 Skills-First Approach

In response to the limitations of traditional hiring practices, the skills-first approach has emerged as a promising framework for selecting the right candidates for specific roles. This

section explores the growing body of literature on the skills-first approach, beginning with a definition, followed by an analysis of the reasoning behind this shift, its benefits, challenges, and practical applications in recruiting. Through this structure, we aim to provide a comprehensive understanding of how skills-based hiring models can reshape candidate evaluation and decision-making in modern recruitment.

3.2.1 Definition of the Skills-First Approach

The skills-first approach is defined in the literature as a talent management model that prioritizes a person's specific skills and competencies over traditional criteria like degrees, job history, or titles. This approach emphasizes directly assessing candidates' abilities for specific roles, focusing on what they can do rather than the formal credentials they hold or how they acquired their skills. By shifting attention to skills, the method is particularly beneficial in industries where practical skills and measurable competencies closely align, such as the digital and technology sectors. The approach aims to democratize access to job opportunities and expand talent pools, enabling organizations to attract, develop, and redeploy talent more inclusively and effectively, particularly in the digital sector (Alhadrami, et al. 2023).

3.2.2 Reasoning for the Skills-First Approach

The skills-first approach marks a significant change in talent evaluation by focusing on candidates' specific abilities as the core of hiring decisions. Supported by extensive research, this approach brings distinct benefits to businesses and individuals, including improved job performance predictions, enhanced workforce diversity, and greater adaptability. The following sections outline the main arguments supporting the concept of skills-first approach in workforce development practices.

Skills as Better Predictors of Job Performance*:* One of the core arguments for the skills-first approach is that skills are more reliable predictors of job performance than traditional

credentials. Research shows that hiring based on a candidate's skills rather than degrees or job titles provides a more accurate measure of their potential to succeed in a role (Hancock, et al. 2022). According to early research from 1984, skills-based hiring is five times more predictive of job performance than education and over twice as predictive as work experience (Hunter and Hunter 1984). Additionally, workers without degrees tend to stay in their roles 34% longer than those with degrees, indicating a retention advantage when hiring based on skills (Hancock, et al. 2022). Supporting these findings, recent research shows that organizations adopting a skills-first culture are 63% more likely to achieve key business and workforce outcomes, including meeting or exceeding financial targets. This insight is based on a Deloitte analysis of the Deloitte Skills-Based Organization Survey conducted in May to June 2022 (Cantrell, et al. 2022). This shift reflects a growing recognition that skills are at least as valuable as formal education in preparing candidates for the demands of modern roles (Sigelman, Fuller and Taska 2024).

Wider Talent Pool and Reduction of Biases: Another key benefit of the skills-first approach is its ability to expand the talent pool and reduce biases inherent in traditional hiring practices. By focusing on skill sets rather than degrees, this approach breaks down barriers commonly faced by candidates without formal education. It opens doors to individuals from varied socioeconomic backgrounds who may have developed relevant skills through alternative pathways, such as online courses, bootcamps, or hands-on experience, rather than conventional academic routes (Fuller, Langer and Sigelman 2022; Hancock, et al. 2022; McRae, et al. 2024). This shift democratizes job access (LinkedIn 2023; Alhadrami, et al. 2023) generating socioeconomic benefits by addressing unequal opportunities, a crucial ethical consideration in education and the hiring process (Tannock 2008).

In addition to socioeconomic benefits, companies also gain from broadening their talent pool to include all skilled workers. Studies show that organizations adopting skills-based hiring can

expand their talent pools by up to ten times, reducing time-to-hire, lowering recruitment costs, and filling roles more quickly (Alhadrami, et al. 2023). Furthermore, companies prioritizing skills over credentials report 36% fewer talent shortages and experience improvements in productivity, work quality, and employee engagement by tapping into this expanded pool of talent (Fuller, et al. 2021).

Overall, by removing the “paper ceiling” (Bankert, et al. 2024) imposed by degree requirements, the skills-first approach empowers employers to build a more diverse, inclusive, and high-performing workforce that fully leverages available talent with diverse competencies (Fuller, Langer and Sigelman 2022; McRae, et al. 2024).

Addressing the Talent Shortage and Meeting Future Demands*:* As organizations face increasing talent shortages and evolving industry demands, the skills-first approach offers a solution by helping to bridge critical skills gaps (Durth et al. 2023). By focusing on key competencies, companies can more effectively meet the demand for skilled workers, particularly in fields where technology and digital proficiency are essential (Fuller, Langer and Sigelman 2022). Technology literacy is one of the most in-demand skills globally, with millions of job vacancies related to digital and tech roles remaining unfilled (Buchanan and Kronk 2022). This growing need for digital skills, which are often not addressed by traditional education alone, has driven many companies to adopt skills-based hiring to capture the competencies required in modern roles (Sigelman, Fuller and Martin 2024).

By focusing on skills, companies can build more robust talent pipelines for the "jobs of tomorrow," supporting long-term adaptability and resilience in an ever-changing workforce landscape (Alhadrami, et al. 2023). Furthermore, organizations that prioritize internal mobility and the development of transferrable skills see increased retention, with employees staying an

average of 5.4 years, which is nearly twice as long as those in companies without such practices (LinkedIn 2022).

Improved Job Matching & Internal Workforce Adaptability*:* In conclusion, the skills-first approach enhances the operations of job matching by emphasizing competency-based selection, allowing companies to identify candidates who excel in the specific skills required for a role (Fuller, Langer and Sigelman 2022). This precision in matching candidates to job requirements, particularly in industries where digital skills are rapidly evolving, enables organizations to align roles with the exact competencies needed for success (Hancock, et al. 2022). Combined with a broader talent pool and the ability to predict job performance more accurately, skills-based hiring supports a more efficient and effective approach to finding the right fit for each position.

With establishing this approach companies can also enhance their internal workforce adaptability by prioritizing candidates with clearly defined learning potential, allowing organizations to quickly adjust to evolving industry demands. By focusing on transferable skills, companies enable employees to transition smoothly into high-demand roles with minimal retraining, improving retention and fostering a more flexible, resilient workforce (Fuller, Langer and Sigelman 2022; LinkedIn 2022).

Category	*Key Factor*	*Relevance to Tech Sector*	*Implications of Skills-First Approach*
Alignment with Job Requirements	*Skills Clearly Related to Jobs*	*Tech roles require specialized, measurable skills (e.g., coding, data analysis).*	*Focuses on assessing technical capabilities directly tied to job performance.*
	Diverse Skills Needed	*Combines hard skills (technical expertise) with soft skills (collaboration, adaptability).*	*Enables holistic assessment of candidates' technical and interpersonal capabilities.*
	High Demand for Specialization	*Skills in AI, cybersecurity, and cloud computing are prioritized over traditional qualifications.*	*Ensures candidates are assessed for critical, role-specific skills.*
Industry Evolution and Innovation	*Fast-Changing Industry*	*Technology evolves faster than formal education.*	*Emphasizes up-to-date, job-relevant skills acquired through alternative pathways.*
	New Roles Without Pathways	*Emerging roles (e.g., AI ethics) lack established degree programs.*	*Evaluates practical knowledge and readiness for innovative roles.*
Workforce Challenges	*Talent Shortage*	*High demand for tech professionals exceeds available talent.*	*Broadens access to nontraditional candidates and untapped talent pools.*
	Remote Work	*Increased remote hiring expands access to global talent pools.*	*Focuses on demonstrated skills rather than geographic proximity or institutional prestige.*
Global Nature of Tech Industry	*Global Hiring*	*Tech is a global industry with diverse talent pools and varying educational systems.*	*Provides universal benchmarks for skills assessment, reducing dependence on inconsistent international credentials.*

Table 1: Suitability of the Skills-First Framework in Digital & Technology-Related Job Fields: *The table highlights how the skills-first framework addresses critical factors in the digital and technology sector. This underscores the framework's suitability for Digital & Technology-Related Job Fields. (Bone, Ehlinger and Stephany 2024; Hancock, et al. 2022; Fuller, Langer and Sigelman 2022; Alhadrami, et al. 2023).*

3.2.4 Barriers to Skills-First Adoption

While there are numerous advantages, its implementation presents several challenges for organizations. These challenges span areas such as skill validation and assessment processes, organizational culture, and the need for stronger support from leadership.

One of the primary hurdles lies in accurately assessing and validating candidates' skills. Many companies lack robust systems to evaluate skills effectively during interviews or through on-the-job performance assessments. This gap underscores the need for a more standardized framework to skills validation to ensure that hiring decisions accurately reflect candidates' competencies (Hancock, et al. 2022).

Another notable challenge, rooted in the absence of standardized frameworks for skills validation, is cultural and managerial resistance. Hiring managers often question whether certain skills can be developed without traditional job experience or degrees, and many find it difficult to redesign interview processes to focus on skills rather than qualifications. Under pressure to fill roles quickly, managers may return to using degrees as a shortcut for assessing candidates' capabilities. This practice reinforces traditional hiring patterns and limits the effectiveness of a skills-first approach (Hancock, et al. 2022; Sigelman, Fuller and Martin 2024).

Moreover, introducing a skills-first hiring approach in a company requires commitment beyond Human Resources involvement. A multi-stakeholder effort is essential to embed these practices effectively across an organization. Without this broad collaboration, establishing skills-based practices across an organization remains challenging. Additionally, it is important to recognize that skills-based hiring may not be suitable for all roles, as certain positions still require formal qualifications (Alhadrami, et al. 2023).

3.2.5 Practical Applications of the Skills-First Approach

Despite the significant potential of the skills-first approach, its adoption is constrained by the implementation challenges mentioned above. While some major companies, such as Google, Delta Airlines, Accenture, and Zoho, have removed degree requirements to broaden their talent pool and focus more on practical skills and experience, the overall shift to skills-based hiring has been slow. Currently, only about 3.6% of roles have eliminated degree requirements, and within this subset, just 37% fully implement skills-based hiring practices. In many cases, companies drop degree requirements in name only, continuing to favour degree-holders over candidates without formal education (Sigelman, Fuller and Martin 2024; McRae, et al. 2024).

However, a growing number of organizations are beginning to see promising outcomes from adopting skills-based hiring practices. For instance, companies that have adopted skills-based job postings report a significant increase in applications from diverse candidates. The Rework America Alliance, which includes prominent employers such as Boeing, Walmart, and IBM, has led initiatives like “Tear the Paper Ceiling” to support skills-based hiring. These companies have removed degree requirements to focus on skills, which has notably improved talent sourcing. One participant in the Rework America Alliance’s Accelerator program increased applications for a role, from just 2 to 12 in a week, ultimately leading to a successful hire (Hancock, et al. 2022).

To further support career growth, some companies use skills-based approaches to establish clear internal career pathways. By mapping future roles and the skills required for advancement, they foster long-term employee development and retention. For example, Siemens has implemented the My Skills platform, an AI-based tool that maps employees' existing skills, identifies gaps, and creates personalized learning paths. This demonstrates how the skills-first approach can be integrated into both hiring and continuous development practices, benefiting both employees and organizations (Moritz and Zahidi 2023).

These cases illustrate that, despite challenges in adoption, the skills-first approach is beginning to show tangible benefits across industries by widening talent pools and supporting employee development. The integration of AI technology, such as Siemens' My Skills platform, further demonstrates the potential for AI to enhance skills-based hiring by providing precise skills mapping, identifying gaps, and creating tailored learning pathways. As organizations continue to adopt the skills-first approach, the strategic use of AI in Human Resource Management (HRM) offers a powerful mean to streamline and scale these practices, maximizing the effectiveness of skills-focused talent strategies.

3.3 AI in Human Resource Management

In this chapter, we examine the transformative role of AI across various HR functions, building upon the Skills-First Approach introduced in the previous chapter. This section explores how AI reshapes HR practices from a traditional administrative focus to a more strategic, data-driven approach. We begin by discussing AI's broad impact on HRM and its integration across the HR lifecycle, and continue with an in-depth look at the specific contributions of AI to modern recruitment practices, challenges in implementation, and the ethical and legal considerations essential for responsible AI adoption in HR. Through this analysis, we aim to highlight AI's potential to enhance efficiency, reduce biases, and improve decision-making across the HR spectrum.

3.3.1 The Transformative Impact of AI on Human Resource Management

AI refers to a broad class of technologies that allow computers to perform tasks that generally require human cognition, such as adaptive decision-making (Tambe, Cappelli and Yakubovich 2019). In HRM the integration of AI represents a transformative shift, moving HRM from a largely administrative function to a strategic partner within organizations. Historically, HRM was primarily focused on administrative duties, such as payroll and record-keeping (Cheng and Hackett 2021). Technological advancements, including the evolution of electronic HRM (e-HRM) and the adoption of AI, have significantly redefined it, enabling more strategic decision-making (Bondarouk, Parry and Furtmueller 2016).

AI-driven HRM systems now assist in various functions such as recruiting, training and development, performance management, and workforce planning. Adoption is influenced by factors such as organizational size, strategic orientation, and technological readiness, with larger organizations often being better positioned to leverage these technologies effectively (Marler and Fisher 2013).

3.3.2 AI Integration Across the Human Resources Lifecycle

The integration of AI into different stages of the HR lifecycle aims to enhance efficiency, reducing manual workloads, and adding precision to HR processes.

One of the primary areas where AI has shown significant impact is **recruitment** (Garg, et al. 2022). AI-driven tools are used to handle high volumes of data associated with candidate screening, facilitating internet-based interviews, assessing applicant fit, and predicting workforce needs. These systems not only speed up recruitment processes but can also reduce biases related to gender and race by structuring candidate evaluations more objectively (Budhwar, et al. 2022). Several large companies have successfully implemented AI driven recruiting tools such as Fetcher or Textkernel's recruiting tool, which scan thousands of job applications and rank candidates, optimizing recruitment efficiency (Fraij and László 2021).

Besides recruiting, AI algorithms are used in **training and development** to track individual employees' skill sets and identify training needs, allowing for the creation of tailored training programs. This personalized approach not only maximizes training effectiveness but also aligns employee growth with organizational goals (Budhwar, et al. 2022).

Furthermore, in **compensation and performance management**, AI-enabled tools help in evaluating employee performance based on multi-faceted criteria rather than subjective human judgment. This leads to more fair assessments, linking performance metrics directly to compensation outcomes (Azadeh, et al. 2018; Manoharan, Muralidharan and Deshmukh 2011). Companies like McDonald's have implemented AI-based tools like Textio to ensure unbiased performance reviews, helping maintain consistency across employee evaluations (Fraij and László 2021).

AI also plays a crucial role in managing **employee well-being and human resource allocation**. AI can analyse engagement data, such as survey results and behavioural metrics, to increase

employee satisfaction and predict turnover risks (Garg, et al. 2022). Systems like XOR, used by McDonald's and IKEA, employ AI-powered chatbots for regular interaction with employees, gathering feedback that can be analysed for well-being indicators and for retention strategies (Fraij and László 2021).

AI's integration across the HR lifecycle has the potential to enhances efficiency and fairness, particularly in recruitment, where it can automate candidate screening, reduce biases, and enable more objective, data-driven assessments. In the next section, we will narrow the focus down to recruiting and highlight the transformative impact of AI on candidate screening.

3.3.3 From Screening to Selection: AI's Impact on Modern Recruitment Practices

AI has fundamentally transformed the recruitment process by providing innovative solutions that streamline every aspect, from initial screening to final hiring decisions. By automating time-consuming tasks such as candidate communication, interview scheduling, and reference checking, manual efforts are reduced, and recruiters can focus on more strategic functions (Fraij and László 2021). This not only enhances efficiency but also improves interactions between recruiters and candidates, improving the overall candidate experience (Niehueser and Boak 2020; Filho 2020).

Focusing on candidate screening and selection, AI brings numerous benefits such as efficiency gains, human bias reduction, and enhanced decision-making.

Improved Efficiency and Cost Savings: AI has significantly enhanced the candidate screening and shortlisting phase by increasing the speed of application processing, enabling recruiters to manage large volumes of applicants more efficiently (Zhang 2024). By leveraging NLP techniques and ML algorithms, résumés can be analysed to extract and evaluate relevant information, streamlining candidate evaluation (Fraij and László 2021).

Practical applications underscore the transformative potential of AI in recruitment. For instance, Hilton Hotels reduced their time-to-hire by 88%, cutting it from 42 days to just 5 (Esch, Black and Ferolie 2019). Similarly, Unilever reduced interview time by 100,000 hours and saved nearly £1 million using AI for résumé filtering and initial assessments (Hocken and King 2023). Additionally, AI-driven pre-screening tools demonstrate impressive accuracy in high-volume recruitment. Hewage (2023) cites a study participant who observed that "every 7 to 8 candidates out of 10 recommended by AI ended up hiring", highlighting the effectiveness of AI in identifying top talent efficiently.

Human Bias Reduction: Another important advantage of AI in recruitment is its potential to minimize unconscious human biases such as those related to race, gender, and age. By relying on data-driven algorithms, AI can systematically evaluate candidates based on quantifiable metrics, thereby promoting objective decision-making (Chen 2023). Cheng and Hackett (2019) mention that "blind screening algorithms are used to eliminate unconscious human bias," fostering fairer recruitment practices. Tools like Google's qDroid help eliminate bias by generating interview questions based solely on job requirements, omitting candidate background details. By concentrating on objective criteria and minimizing subjective judgments, AI contributes to more equitable hiring decisions, promoting diversity and inclusion within organizations (Fraij and László 2021).

Enhanced Decision-Making: Finally, AI in recruiting enhances decision-making by providing HR professionals with objective, data-driven insights that improve the fairness and consistency of candidate evaluations (Zhang 2024). Delecraz et al. (2022) propose an algorithm focusing on various competency dimensions, ensuring candidates are selected based on job-relevant capabilities rather than demographic or unrelated characteristics. This competency-focused approach improves candidate-job fit by concentrating on relevant skills and experiences and

enables HR teams to make well-informed decisions based on quantifiable metrics rather than subjective judgment alone (Zhang 2024).

3.3.4 Challenges of AI in Recruiting and Selection

While AI offers numerous benefits in recruitment, it also presents several challenges that must be addressed to ensure its effective use. These challenges include system complexity, lack of decision transparency, employee resistance, and the limitations of algorithmic performance.

Complexity of Integration and Data Quality Concerns: AI systems require extensive data collection, preprocessing, and model training, which can be resource-intensive and time-consuming. Smaller organizations may lack the necessary infrastructure and expertise to implement AI successfully, making it difficult for them to achieve the same benefits as larger companies. Additionally, the effectiveness of AI models is generally highly dependent on the quality of data, and poor data quality can lead to unreliable outcomes (Tambe, Cappelli and Yakubovich 2019).

Employee Resistance: The fear of job displacement and concerns over technological disruption can lead to scepticism towards AI adoption. Employees and hiring managers may be wary of relying on AI for decision-making, particularly when it comes to assessing human qualities that are inherently complex and subjective. Proper training, communication, and fostering a culture that positions AI as a tool to assist rather than replace human roles are critical for reducing resistance (Zhang 2024; Budhwar, et al. 2022).

Balancing AI Efficiency with Human Judgment in Recruiting: While AI can efficiently handle a variety of tasks such as screening résumés and ranking candidates, human judgment is essential for evaluating nuanced qualities like cultural fit, team compatibility, and interpersonal skills. Relying solely on AI may overlook these critical aspects of candidate evaluation. Therefore, it is crucial for organizations to strike a balance between AI-based

assessments and human input to ensure a well-rounded recruiting process (Fraij and László 2021).

In summary, while AI holds great potential for transforming recruiting processes, several challenges must be addressed to ensure its effective and responsible use. Overcoming these requires a combination of technological improvements, organizational readiness, and a balanced approach that integrates human oversight into AI-driven recruiting practices.

3.3.5 Ethical and Legal Considerations in AI-Driven Recruiting

Building on the challenges of AI in recruiting, it is essential to investigate deeper into the ethical and legal considerations since they have significant implications for the effectiveness and acceptance of AI-driven recruitment. Addressing these concerns is crucial for ensuring fair and responsible hiring practices.

Algorithmic Bias and Fairness: One of the major ethical challenges in AI-driven recruiting is the risk of algorithmic bias (Caliskan, Bryson and Narayanan 2017). Early AI applications, such as Amazon's hiring algorithm, revealed biases against women and people of colour due to the data used for training, which reflected past discriminatory practices (Fraij and László 2021). Reducing such biases is critical to promoting equity, diversity, and inclusion in AI-driven recruiting. The use of biased data can also have serious reputational and legal consequences for organizations, making bias mitigation a priority for responsible AI deployment (Raghavan, et al. 2020).

Transparency in AI Decision-Making: Many AI algorithms operate as "black boxes," making it difficult for HR professionals and candidates to understand how decisions are made (Cheng 2021). This opacity can result in mistrust from both recruiters and candidates, who may feel uncomfortable with decisions made by algorithms that they do not understand. Transparency is

crucial for gaining acceptance of AI-driven recruiting, and organizations need to find ways to make AI processes more explainable and understandable (Delecraz, et al. 2022).

Regulatory Challenges and Privacy Concerns: Regulatory and legal and issues are also significant considerations when deploying AI in recruiting. AI accountability and data protection, particularly in regions governed by regulations like the GDPR, are crucial for maintaining ethical standards (Budhwar, et al. 2022). Ensuring that AI systems comply with data privacy laws, such as the European Commission's AI Act, helps protect candidate rights and promotes responsible use. This legislative framework emphasizes transparency and mandates that certain AI systems explain their decision-making processes, which can help alleviate public concerns about AI use in recruitment (Council 2024). Moreover, organizations must implement stringent data security measures and ensure that only necessary information is collected and processed to minimize the risk of privacy violations (Hunkenschroer and Luetge 2022).

In summary, ethical and legal considerations in AI-driven recruiting encompass issues of bias, transparency, data privacy, and accountability. Addressing these ethical and legal challenges is essential for building trust, ensuring fairness, and complying with legal standards in recruitment practices. By focusing on responsible AI deployment and integrating ethical guidelines into AI systems, organizations can enhance the fairness and inclusivity of their recruiting processes while mitigating risks associated with AI use.

3.4 Technical Perspective on Matching Models in Recruitment

This section provides a technical perspective on the evolving landscape of matching models and their applications in recruitment, focusing on market trends and advanced computational solutions. The analysis begins with the theoretical foundations of matching models, including explicit and implicit matching methodologies, and their relevance to recruitment processes. The

role of recommender systems (RS) in candidate-job matching is then explored, highlighting content-based filtering as a suitable technique for skills-based matching. Additionally, the section traces the evolution of matching models from manual approaches to AI-driven solutions, emphasizing the integration of ML and NLP. This analysis explains recent advancements, provides a side-by-side comparison of state-of-the-art models and outlines future trends and challenges in applying skills-based candidate-job matching in real-world scenarios.

3.4.1 Theoretical Foundations of Matching Models and Recommender Systems

In the following section we outline the theoretical foundations of matching models and recommendation systems, both of which underlie our candidate-job matching approach. Our model exhibits characteristics of implicit matching and follows a content-based approach, leveraging NLP techniques and ML models to assess the alignment of candidate skills with job requirements.

Matching Models: A matching model is a computational algorithm designed to pair elements from two distinct groups based on defined criteria or preferences, aiming to create optimal or stable matches. In computer science, these models are commonly used to match agents, such as job seekers with employers, by analysing factors like preferences, skills, and attributes to identify the best possible pairings. These models are fundamental in various applications, including recruitment systems, college admissions, organ donations, and ride-sharing platforms (Ren, et al. 2021). The concept of matching models, particularly in the form of two-sided matchings, was first introduced by Gale and Shapley (1962) in their seminal paper “College Admissions and the Stability of Marriage.” They formulated the college admissions problem and proposed the Deferred Acceptance Algorithm to find stable matchings between two sets of agents with preferences over each other. Since then, the theory of two-sided matching and its

applications have been extensively developed in the economics literature (Roth and Oliveira Sotomayor 1990).

Matching models are broadly categorized into two types based on how preferences are accessed and utilized: explicit matching and implicit matching (Ren, et al. 2021).

In **explicit matching**, agents on both sides provide their own preference lists. Each agent ranks the agents on the opposite side based on their preferences, and matching algorithms use these rankings to find an optimal match. The goal is to reach a stable state where no pair of agents would prefer to be matched with each other over their current match. The different types of explicit matching are One-to-One, Many-to-One and Many-to-Many matching (Ren, et al. 2021).

In contrast, **implicit matching** infers agents' preferences indirectly by calculating matching scores based on object similarities, without explicit preference lists. This approach evolved due to the impracticality of obtaining explicit preferences in large-scale applications, especially with big data. Types of implicit matching include Retrieval Matching and User-Item Matching. Retrieval Matching, also known as question-answer matching, matches a user's query to relevant documents by calculating similarity scores, as seen in search engines and question-answering systems. User-Item Matching, prevalent in recommendation systems, recommends items to users by inferring preferences from user behaviour and item attributes, common in e-commerce and streaming services (Ren, et al. 2021).

In the context of this project, we develop an implicit matching model that is closely related to retrieval matching, as it involves matching résumés to job descriptions by calculating similarity scores based on extracted skills, mirroring how retrieval systems match user queries to relevant documents in information retrieval.

Recommendetion Systems: Recommendation systems are algorithms developed to help users navigate large volumes of information by predicting their preferences or ratings for specific items (Adomavicius and Tuzhilin 2005). These systems play a crucial role in scenarios where users face an overwhelming number of choices, such as in e-commerce or media streaming, by narrowing down options based on inferred preferences, thus enhancing user experience. By providing personalized recommendations, recommendation systems allow users to find relevant content or products more efficiently, which has led to their widespread adoption across various digital platforms (Ricci, Rokach and Shapira 2021). Recommendation systems were first introduced in the academic literature by Paul Resnick and Hal R. Varian in their seminal paper titled “Recommender Systems” published in 1997. This paper laid the groundwork for understanding and classifying RS, describing their potential to help users make decisions by filtering and prioritizing information based on preferences (Resnick and Varian 1997).

Recommendation systems are typically classified into two main types: content-based recommendation systems and collaborative filtering systems, each employing distinct methods to generate suggestions (Adomavicius and Tuzhilin 2005; Tejaswini, et al. 2021).

Content-based recommendation systems use data that the user provides, either directly through ratings or indirectly through browsing and search behaviour, to build a user profile. The system then generates recommendations by matching this profile with item characteristics, suggesting content with similar features to those the user has previously shown interest in. As user input accumulates over time, content-based systems become increasingly accurate, personalizing recommendations based on item attributes that align with user preferences (Adomavicius and Tuzhilin 2005; Tejaswini, et al. 2021).

Collaborative filtering, in contrast, relies on user collaboration by generating recommendations based on collective behaviour patterns and preferences among users. It

identifies users with similar tastes and suggests items that others with comparable preferences have enjoyed. Collaborative filtering is effective for discovering new items that users might not have encountered based solely on their profile, as it leverages the preferences of like-minded individuals rather than focusing on item characteristics alone (Adomavicius and Tuzhilin 2005; Tejaswini, et al. 2021).

In this paper, the concept of content-based filtering is used for recommendation purposes. This approach is particularly suitable for recruitment as it enables precise matching of candidate skills with job requirements by focusing on explicit attributes within résumés and job descriptions, aligning well with the skills-first hiring approach.

Strengths and Limitations of the Content-Based Approach: The content-based approach has notable strengths and limitations, particularly when applied in recruitment contexts. Content-based filtering is well-suited for handling both structured and unstructured data, making it ideal for analysing résumés and job descriptions where specific skills and qualifications can be directly matched to job requirements (Freire and de Castro 2021). By focusing on explicit attributes of candidates and job postings, content-based recommendation systems enable precise alignment with the desired skills and experience for each role, supporting a skills-first hiring framework that emphasizes relevant qualifications over traditional markers.

However, content-based recommendation systems rely heavily on accurate feature extraction and representation; inconsistencies or errors in skill or job descriptions can undermine recommendation accuracy (Freire and de Castro 2021). This reliance on well-defined attributes can be challenging in recruitment, where résumés and job descriptions may vary in detail and format. Additionally, content-based systems often struggle in cases where candidates or jobs lack sufficient descriptive data, leading to difficulty in generating meaningful recommendations when profiles are incomplete or lack standardization.

Under consideration of both the advantages and challenges of content-based filtering, this paper aims to apply this approach to improve candidate-job matching by prioritizing skill alignment and data accuracy.

3.4.2 Evolution of Candidate-Job Matching Techniques

Understanding how candidate-job matching techniques have evolved is crucial for understanding the advancements and limitations of modern recruitment systems. Before technology was used within recruiting, paper résumés and cover letters were manually reviewed in a time-consuming and inconsistent process (Holm 2012). The 1980s introduced digital databases, enabling keyword and Boolean searches to streamline candidate filtering. However, these methods often produced inaccurate results due to keyword manipulation and variations in language (Kavanagh, Gueutal and Tannenbaum 1990; Strohmeier 2007).

By the late 1980s and early 1990s, Applicant Tracking Systems (ATS) emerged, automating the storage and basic filtering of applicant data. This innovation significantly improved the efficiency of handling large volumes of applications (Kavanagh, Gueutal und Tannenbaum 1990) (Chapman 2003). In the mid-1990s, one of the first job recommendation systems was introduced – a Minitel-based service that used keyword matching and fixed job taxonomies to match résumés to job descriptions (Freire and de Castro 2021).

The early 2000s marked advancements in NLP, which enabled basic résumé parsing to extract structured data. While this innovation streamlined the recruitment process, limitations in understanding language complexity continued to hinder precise candidate-job matching (Chowdhury 2005; Lavelli 2004).

The integration of ML and AI in recruitment gained momentum in the mid-2000s and the 2010s, revolutionizing the field with approaches like semantic similarity-based job matching and data-driven decision-making (Freire and de Castro 2021; Garg, et al. 2022). Initial efforts focused

on enhancing e-recruitment functionality through early recommendation system models (Al-Otaibi 2012) while more recent studies introduced learning-based solutions that predict matches by analysing past cases (Martinez-Gil, et al. 2020). Advanced AI techniques now play an important role in text analytics and NLP for automated candidate-job matching (Rojas-Galeano, et al. 2022).

State-of-the-art systems employ embedding-based recommendation systems for large-scale matching, fusing raw text, semantic entities, and location information. These models achieve high accuracy, efficiency, and scalability, addressing the information overload often associated with online recruitment platforms (Zhao, et al. 2021). This technological progression has transformed candidate-job matching into a sophisticated, data-driven process, setting the stage for ongoing innovations.

3.4.3 Current Developments in Skills-based Candidate-Job Matching Models

Building on the historical evolution of candidate-job matching systems, recent advancements emphasize skills-based approaches. Reviewing skills-based models from recent literature provides valuable insights into the techniques, algorithms, and methodologies used for skill extraction, representation, and alignment. These insights lay the foundation for constructing our own skills-based matching model.

Petrican et al. (2017) introduced an ontology-driven skills-based matching model that organizes skills hierarchically within an ontology. This approach allows similarity calculations based on the "distance" between skills within the hierarchy. Their model explores three algorithms for skill matching: Exhaustive Search, Breadcrumbs, and Multi-Source Shortest Path (MSSP). MSSP, the most effective of the three, treats the ontology as a graph, using a modified shortest-path algorithm to match skills within a predefined distance. This method highlights the importance of structured representations for enabling skill comparisons (Petrican, et al. 2017).

In 2022, Konstantinidis et al. developed the Resume2Skill-SE model, which uses Siamese BERT Networks and MPNet to embed résumé content and ESCO skill descriptions. These embeddings are indexed with Facebook AI Similarity Search (Faiss) to enable rapid similarity calculations. The model incorporates a Profile-Skills Bipartite Graph that connects résumé nodes to skill nodes, with weighted edges reflecting skill relevance. This graph-based structure refines candidate-job alignment by integrating mention frequency and other contextual factors. Evaluation metrics such as Mean Average Precision (MAP) and Average Discounted Cumulative Gain (ADCG) were used to assess the model's ranking precision (Konstantinidis, et al. 2022).

Kara et al. (2023) proposed a job recommendation system that employs Word2Vec embeddings for skill representation, focusing on semantic relationships within a predefined skill dictionary. For similarity scoring, their model uses Word Mover's Distance (WMD) to calculate semantic "cost" between skill vectors from résumés and job descriptions, offering a nuanced method for skill matching. Additionally, they incorporated cosine similarity with Inverse Document Frequency (IDF) weighting to prioritize rarer, critical skills. Comparative testing demonstrated WMD's superior performance in nuanced matching tasks (Kara, et al. 2023).

Most recently, Pundir et al. (2024) introduced a skills-first résumé recommendation system integrating Word2Vec embeddings and Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) to analyse sequential résumé data. This approach enables dynamic career path analysis alongside skills-based matching. The model applies cosine similarity with weighting mechanisms to emphasize key skills, offering both static and dynamic matching capabilities. By combining embedding models with deep learning, this system provides enhanced candidate-job alignment and predictive capabilities for career progression (Pundir, et al. 2024).

The reviewed models demonstrate diverse methodologies for skill extraction, representation, and alignment. Ontology-based structures, graph-based frameworks, advanced semantic metrics, and deep learning techniques highlight the range of tools available for tackling the complexities of skills-based matching. These methods achieve varying balances of computational efficiency, precision, and scalability. Building on these insights, we built our own skills-based matching model leveraging an extensive skill-dictionary and state-of-the-art embedding models for semantic skill alignment to balance simplicity and effectiveness that we will outline in detail in Section 5.

3.5 Summary of Literature Findings

Recent research highlights the limitations of traditional hiring criteria, including the overreliance on degrees and experience, which often exclude diverse and skilled candidates. This has highlighted the potential of the skills-first approach, which shifts the focus toward measurable skills and competencies, particularly in fields like digital and technology, where practical abilities are crucial. However, adopting this approach faces several hurdles, such as challenges in skill validation, organizational resistance, and the need for cultural shifts within hiring processes.

While advanced hiring algorithms are now widely used in business operations, their implementation raises concerns about algorithmic bias and ethical considerations. Skills-first AI-driven matching models present a promising alternative by reducing biases through objective assessments focused only on candidates' skillsets. Despite this potential, technical solutions for candidate-job matching remain fragmented, with no universally accepted methodology. Early models that incorporate the skills-first approach demonstrate promise but lack standardization and scalability highlight opportunities for further development and refinement.

4. Skill Exploratory Data Analysis

As outlined in the literature review, a skills-first strategy prioritizes assessing candidates based on their skills and competencies rather than traditional qualifications such as degrees or job titles. The skills-first approach has the potential to expand the pool of relevant candidates in external recruitment (Alhadrami, et al. 2023) and improve internal workforce adaptability by promoting role mobility, enabling organizations to respond swiftly to changing industry demands. Central to this approach is the assumption that different roles often share overlapping skill sets. For instance, it would only be practical to recruit a data analyst for a data scientist role, or to support an internal transition between these roles, if they possess similar competencies. If such overlap exists, focusing exclusively on skills to identify suitable candidates would be more effective than limiting the search to those with direct experience in the same role. This analysis seeks to answer the following hypothesis: *"Technological roles exhibit substantial skill overlap, enabling role transitions and supporting a skills-first approach in recruitment and workforce development."*

The analysis follows three main steps. First, the **degree of overlap between different roles** is investigated. This phase includes an examination of specific skill differences for some roles to provide a deeper understanding of the level of overlap and the ease with which individuals can transition between positions. Second, we closely evaluate the **extent of overlap and transitions between roles**, examining whether the ability to move between roles is influenced by additional factors. Specifically, we analyse how industry-specific skill requirements impact the degree of role overlap. The final step of our analysis addresses **the limitations of our initial findings** and seeks to validate the results by applying a similarity analysis to an independent dataset.

4.1 Data Collection

The initial phase of this research focuses on identifying a dataset that could provide meaningful insights into the skills and career trajectories of individuals in technological-related roles. After evaluating several potential sources, the **Kaggle Survey Dataset** from 2022 (Mooney 2022) was chosen, which offers comprehensive information on a wide range of professional profiles, including their education, job roles, tools, and skills.

The Kaggle Survey is an industry-wide survey that presents a comprehensive view of the state of data science and ML (Mooney 2022). The dataset comprises 23,998 rows and 296 columns. A significant portion of the columns relate to questions about various skills, most of which are presented in a multiple-choice format. For example, one question addresses the data visualization libraries regularly used, allowing respondents to select from 13 different options. In addition to skills, the dataset provides information on respondents' educational backgrounds, job roles, industries, and countries of residence.

4.1.1 Datasets Considered

Several datasets were considered for the analysis. These included other datasets from Kaggle about AI and ML jobs in the US, and job boards such as Dice.com or Glassdoor. While these platforms provide detailed information on professional backgrounds and job descriptions, they often lack the structured format necessary for the scope of this study, making them less suitable for systematic analysis. In contrast, the Kaggle Survey offers several advantages that make it suitable for this research. First, the diversity of data ensures a broad representation of respondents from various geographic regions, industries, and roles, contributing to a comprehensive understanding of the technology labour markets. Second, the dataset's structured format provides well-organized data points on individual skills (e.g., programming

languages and software tools) and job roles. Lastly, the dataset is highly relevant to the research focus due to its emphasis on tech-related roles.

4.1.2. Limitations of the Dataset

While the Kaggle Survey provides a rich source of information, it also presents several limitations that need to be considered. First, the dataset does not provide direct measurements of skills. Instead, it poses questions such as *"What programming languages do you use on a regular basis?",* which are then used as a proxy for assessing skills. As a result, the dataset may not accurately reflect the depth or quality of respondents' competencies. Additionally, the resulting data frame suffers from unbalanced representation. Certain roles, such as Data Analysts, are overrepresented, while others, like Researchers, have insufficient data. There is also a notable skewness toward certain industries or regions, with 30% of respondents working in the Computers/Technology industry, compared to only 3.39% in the Insurance & Risk Assessment industry. This imbalance makes it difficult to accurately determine the skill requirements for different positions and to generalize findings across industries. Another limitation is the reliance on self-reported skill levels, which can introduce biases and inconsistencies. Respondents may rate their skills in relative terms rather than absolute ones, leading to potential discrepancies. For example, an ML Engineer might not select random forests as a regularly used tool, even if their knowledge of the algorithm is more advanced than that of a Data Scientist who did select it. This variation in how individuals report tool usage can complicate direct skill comparisons across roles.

In summary, despite these limitations, the Kaggle Survey dataset was chosen for its comprehensive coverage and relevance to the research objectives. For a more robust analysis, future studies could benefit from more refined and balanced datasets that include validated skill assessments and detailed job descriptions.

4.1.3. Feature Selection

Given the dataset's complexity, consisting of 44 questions, many offering more than seven response options, cleaning and feature selection were essential before conducting any analysis. Data cleaning involved removing roles outside the scope of the analysis, Nan values for the Role column and creating dummy variables. More details are given in Appendix A.1.1. After cleaning, the resulting DataFrame consisted of approximately 9,000 observations.

Initially, features were chosen by their potential to serve as proxies for skills. As a starting point, all questions related to activities and tools that respondents regularly perform were included, reflecting practical, skill-related behaviours. Following this, a correlation analysis was conducted to identify any important features that may have been overlooked. However, since no features with strong correlations were found, no additional features were included. In Appendix A.1.2, the formula and results of the correlation are shown in more detail.

The final set of selected skills encompassed the following key features: (1) Programming Languages, (2) Integrated Development Environments (IDEs), (3) Data Visualization Libraries, (4) ML Algorithms, (5) NLP methods, (6) Primary Activities related to the role, (7) Years of Programming Experience, (8) Job Role, and (9) Industry. After being transformed into dummy variables, the dataset expanded to a total of 111 columns.

4.2 Role Overlap

This section aims to identify which roles share a similar set of competencies and examine the degree of skill overlap between roles. Analysing these aspects will provide information about the potential for role mobility and enable the identification of areas where upskilling or reskilling may be necessary. An important note to highlight is that the focus is on tech-related roles, as these are most relevant to this study. For this reason, the roles selected are Data

Scientist, Data Analyst, Software Engineer, Research Scientist, ML Engineer, and Data Engineer.

4.2.1. Quantify the Skill Overlap Between Roles

To measure the degree of overlap between roles, different similarity metrics are used as they provide a systematic way to quantify the extent to which different skill sets align. Several common similarity measures (Sohangir and Dingding 2017) were considered, including:

1) Euclidean distance which measures the straight-line distance between two points in a multi-dimensional space, focusing on the magnitude of differences between skills,

$$\text{Euclidian distance } = \sqrt{\{(x_2 - x_1)^2 + (y_2 - y_1)^2\}} \quad (1)$$

2) Jaccard similarity that captures the proportion of shared elements between two sets relative to their union, ideal for binary data representation,

$$\text{Jaccard Similarity } (A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

3) Cosine similarity which assesses the cosine of the angle between vectors, measuring directional alignment and relative emphasis on skills,

$$\text{Cosine Similarity } = S_C(A, B) \coloneqq \cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (3)$$

4) Hamming distance that evaluates the number of positions at which corresponding elements are different.

$$\text{Hamming Distance} = \sum_{\{i=1\}}^{n} (x_i \neq y_i) \quad (4)$$

Initially, we used Jaccard similarity to measure the similarity to maintain the binary nature of the original data, allowing for the calculation of similarity between binary vectors before

aggregating. The process involved computing pairwise similarities for each observation within role pairs and then averaging the results. However, this approach proved to be computationally intensive and ultimately impractical for large-scale analysis. Therefore, we explored a second method that involved aggregating the raw data from the survey to get a binary vector per each role and then calculate the Jaccard similarity. The binary vectors were applied by using a threshold based on skill usage - assigning a binary value of 1 if at least X% of individuals of a role possessed a skill, and 0 otherwise. This method became problematic as similarity scores varied greatly depending on the chosen threshold, making it hard to balance key skill identification without losing valuable information.

Ultimately, we recognized that the most effective approach for our analysis involved using cosine similarity and euclidean distance which are typically used for continuous data. Both methods are widely used distance measures in a wide range of applications from NLP to clustering and recommendation systems (Heidarian and Dinneen 2016; Kumar, Chhabra and Kumar 2013; Kużelewska 2014). The binary features were transformed into a continuous scale by using the same approach as mentioned above, calculating the mean skill usage for each role.

Cosine similarity captures the cosine of the angle between two vectors, focusing on their directional alignment irrespective of magnitude. As an example, consider two roles, Data Scientist and Research Scientist possessing two skills, Python and SQL. Even if the Data Scientist role reports a higher average usage level of Python (e.g., 50%) compared to a Research Scientist with lower usage (e.g., 6%), their cosine similarity can still be high if the relative usage pattern between Python and SQL, is consistent across both roles. This is because cosine similarity focuses on the direction of the vectors - reflecting the relative usage among different skills - rather than the absolute usage values.

Euclidean distance, in contrast, calculates the straight-line distance between two points in space and considers both the direction and magnitude of the vectors. This means that, unlike cosine similarity, euclidean distance will reflect greater disparity between roles if the overall magnitude of their skill usage differs, even if the proportional relationships between skills are the same.

Both methods provided valuable insights, so we opted to continue with both in our analysis. However, one limitation to note is that the similarity measures are based on percentage data constrained to positive values between 0 and 1, positioning the skill vectors in the northeast quadrant of the Cartesian plane. As a result, the expected mean cosine would converge towards 0.75 as we increase the number of dimensions. The reasoning is detailed in Appendix A.2.1. This restriction limits the range of potential distance outcomes and results in generally higher similarity scores.

Findings and Implications for a Skills-First Approach: The results of both the cosine similarity and euclidean distance measures reveal a consistently high degree of similarity among all roles, underscoring a significant overlap in skill usage across technology-related positions. Specifically, the cosine similarity scores range from 0.84 to 0.97, indicating that the roles share similar patterns in the relative usage of skills - the directions of their skill usage vectors are closely aligned. Complementing this, the euclidean distances between roles range from 0.61 to 1.88, on a scale where 0 signifies identical roles. In theory, maximum euclidean distance is unbounded, but in practice, if the input values are bounded between zero and one, the theoretical maximum is equal to the square of the features. With 78 features in the dataset, the maximum distance is approximately 8.81. These relatively low distance values confirm that the roles are not only similar in the direction of their skill usage (as shown by cosine similarity) but also in the magnitude of that usage. Since both metrics yield similar results, euclidean

distance will be used as the primary similarity measure in the following examples, as it captures both the magnitude and direction of skill usage.

The results indicate that the lowest observed distance is between Data Engineer and Software Engineer, with a euclidean distance of 0.61. Conversely, the Research Scientist and ML Engineer roles exhibit the highest distances with other roles, suggesting more distinct and specialized skill sets. This can be attributed to the theoretical nature of the Research Scientist role, which often requires domain-specific expertise. In contrast, ML Engineers possess a specialized skill set focused on complex algorithm development and system implementation, resulting in an advanced but more narrow profile. Additionally, Data Engineers and Data Scientists demonstrate higher overlap with many roles, indicating a broad and adaptable skill set applicable across multiple positions within the tech industry. Out of the six roles analysed, four of them share the same top overlapping role across both methods (euclidean distance and cosine similarity), indicating strong similarities in both the relative and absolute usage of skills. The top overlapping roles based on the euclidean distance are given in the Figure 2, with its corresponding values. Scores for both methods are available in Appendix A.2.1.

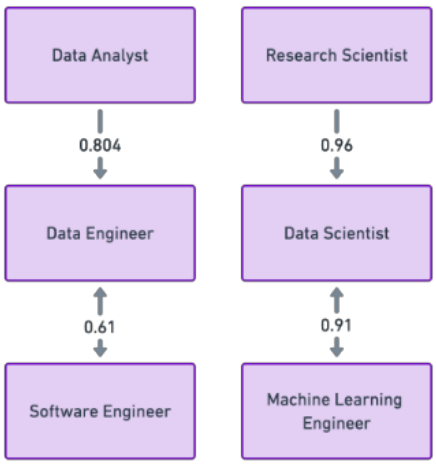


Figure 2: Lowest euclidean distances: *The graph shows the similarity between roles, with arrows indicating the direction of the closest relationship; a double arrow represents mutual similarity. For example, a Data Analyst's closest role is a Data Engineer, while a Data Engineer's closest role is a Software Engineer, which in turn has Data Engineer as its closest role.*

These findings have important implications for the feasibility of a skills-first approach in recruitment and workforce development. The substantial overlap in skills among various tech-related roles supports the notion that focusing on specific competencies rather than traditional hiring criteria, such as job titles, can expand the pool of qualified candidates. For example, professionals in roles like Data Analyst or Software Engineer may possess the requisite skills to perform effectively in a Data Scientist position, thereby enhancing flexibility in hiring and career transitions.

However, it is important to acknowledge the limitations of the analysis. The constraints of using positive-only input values, combined with high-dimensional feature space, may have influenced the generally higher similarity scores observed. Consequently, while the observed similarities are significant, they may somewhat overstate the extent of overlap due to these methodological factors.

4.2.2. A Closer Examination of Overlaps Through Skill Distribution Plots

This section provides a more detailed examination of role overlaps by analysing skill distribution plots. While the similarity analysis from the previous section suggests that most roles share many skills, it is essential to examine the actual distribution of these skills to understand their practical implications. Specifically, we aim to identify any potential gaps between roles with high similarity scores and explore how these gaps impact the effectiveness of a skills-first approach.

To illustrate the distribution of skills between roles, we examine the skill disparities between selected pairs: **Software Engineer & Data Engineer** and **ML Engineer & Data Scientist**. These pairs were chosen for two reasons: 1) Each role in the pair was top ranked in similarity to the other based on similarity measures, 2) The similarity results were consistent across both euclidean and cosine similarity metrics, reinforcing their reliability for detailed analysis. The

objective of focusing on these specific role pairs is not to provide detailed information exclusively about these roles, but rather to uncover the nuances in skill overlaps on a broader level. By doing so, we can address questions such as: Are there significant skill gaps despite high similarity scores? Do some roles necessitate additional training or development for effective transitions? What are the broader implications of these findings for the application of a skills-first approach in recruitment and workforce development?

To detect significant differences in skills between paired roles, a chi-squared test of independence was applied. The measure was selected because it is well-suited for analysing relationships between categorical variables - namely, the dependent variable (Role) and the independent variables (Skills) - without requiring the data to follow a normal distribution (McHugh 2013). The chi-squared test compares the frequency of specific skills between two role groups (e.g., Software Engineer and Data Engineer) by calculating the chi-squared statistic and corresponding p-value (Plackett 1983).

Overlap Between Software Engineer & Data Engineer: In total, there are 30 skills where usage differs significantly. Focusing on the core skills - those with usage equal to or above 50% in either role - we see six statistically different skills where the Data Engineer role has higher usage in all of them. Among daily activities, the largest disparity is in building data infrastructure, with 52.8% of Data Engineers performing this task regularly compared to only 23.1% of Software Engineers. The most pronounced gap among more technical related skills is in SQL usage, where 71.5% of Data Engineers use it regularly versus 52.9% of Software Engineers, a difference of 18.6 percentage points. These findings suggest that while both roles share a solid technical foundation, Data Engineers place a greater emphasis on data infrastructure and management skills. This specialization means that transitioning from Software Engineer to Data Engineer may require targeted reskilling and upskilling in certain

areas. Conversely, Data Engineers may find it smoother to transition to Software Engineer roles, as the areas where they excel are supplementary to software engineering tasks.

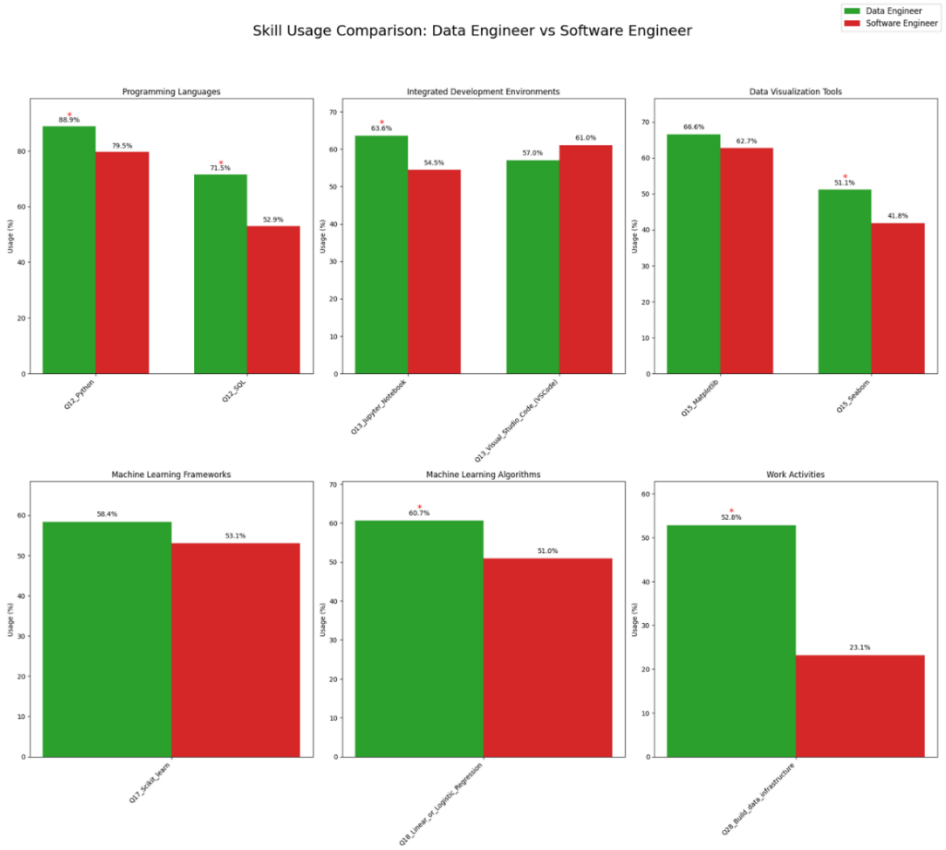


Figure 3: Skill Usage Differences Between Data Engineer & Software Engineer: *The image compares the skill usage of these roles across six categories. Each bar chart shows tools used within a skill, with percentages representing the proportion of respondents using each tool. Red dots indicate significant differences in usage between the two roles.*

Overlap Between Data Scientist and ML Engineer: The skill comparison between Data Scientists and ML Engineers reveals notable differences that affect their role overlap. Among the 50 significantly different skills identified, ML Engineers have higher average usage in eight of the 15 core skills. In terms of daily activities, Data Scientists focus more on analysing data, while ML Engineers are more engaged in ML product workflows, indicating a stronger emphasis on deploying and operationalizing models. Key disparities lie in ML models and frameworks. Data Scientists more frequently use traditional ML techniques like linear and logistic regression, decision trees, and gradient boosting machines. In contrast, ML Engineers are more proficient in advanced deep learning methods, notably convolutional neural networks (CNNs): 62.4% of ML Engineers use CNNs compared to 36.6% of Data Scientists. These

divergences imply that transitioning between the two roles may require targeted upskilling - Data Scientists moving toward ML Engineering may deepen their expertise in advanced deep learning techniques and operational tools, while ML Engineers aiming for Data Science roles might focus on enhancing their skills in data analysis and statistical modelling.

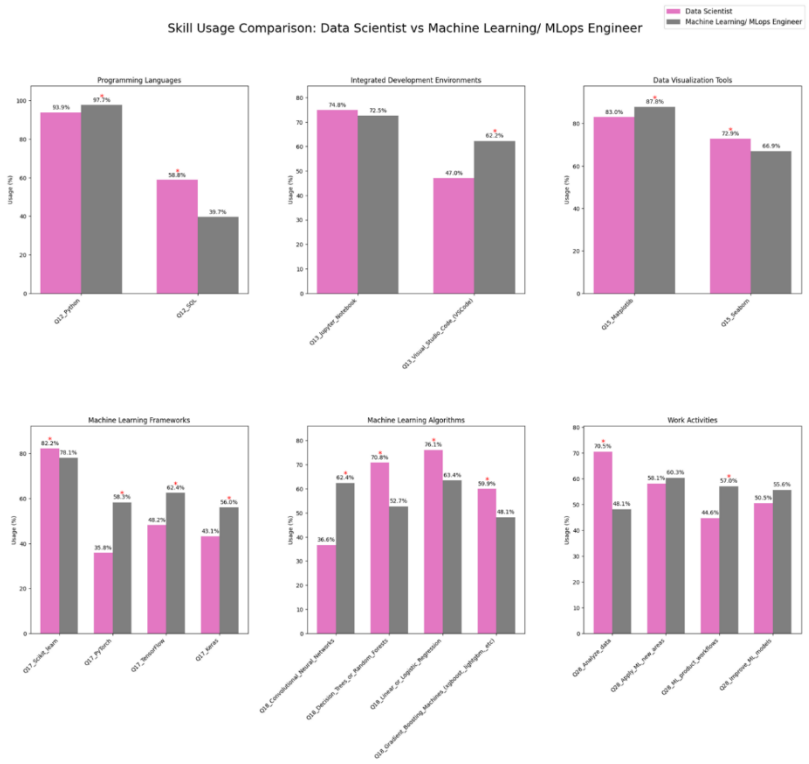


Figure 4: Skill Usage Differences Between Data Scientist & ML Engineer: *The image compares the skill usage of these roles across six categories. Each bar chart shows tools used within a skill, with percentages representing the proportion of respondents using each tool. Red dots indicate significant differences in usage between the two roles.*

In summary, the analysis above highlights notable skill gaps between the roles, indicating that the cost and effort involved in transitioning can vary. A skills-first approach can facilitate cross-role mobility, but it must be accompanied by tailored development plans that address specific skill deficiencies. This ensures that individuals and organizations can effectively manage career progression and talent development, optimizing both adaptability and expertise within the tech sector.

4.3 Industry Context and Its Impact on Skill Overlap

With the most similar roles identified and their degree of overlap established, the next step is to investigate whether this overlap is influenced by industry-specific skill requirements. The analysis aims to identify sectors where skill alignment enhances workforce flexibility and candidate selection, as well as those that require specialized competencies, which may limit role adaptability and reduce the effectiveness of a skills-first approach.

To spot industry differences more effectively, we examined how the euclidean distance between pairs of roles changes across industries. The results can be found in the Appendix A.3. Overall, our observations indicate that the euclidean distances between role pairs remain relatively low across different industries, ranging from 0.64 to 2.76 among all role pairs. This suggests that, in general, there is a high degree of overlap in skills between these roles across various sectors.

However, certain sectors consistently exhibit higher euclidean distances, reflecting specialized skill requirements that make transitions more difficult. For example, Data Analyst and Data Engineer exhibit greater similarity in the Computers and Technology sector (euclidean distance of 0.72) compared to the public sector (euclidean distance of 1.43). When considering specific industries, the similarity score changes the overall score of 0.8, illustrating how industry context influences role compatibility. In technology-driven industries, overlapping skills and shared methodologies make transitions between these roles more seamless. In contrast, the specialized requirements in the public sector create greater skill gaps, making role mobility relatively more challenging. Our analysis also highlights that industries like Insurance & Risk, Online Services, and Government & Public Sector consistently give the highest euclidean distances across all role pairs. This suggests that transitioning between roles in these sectors may require additional upskilling or reskilling. Conversely, the Computers and Technology industry consistently

shows the lowest euclidean distances, indicating that roles within this sector share more similar skill requirements, making career transitions more feasible.

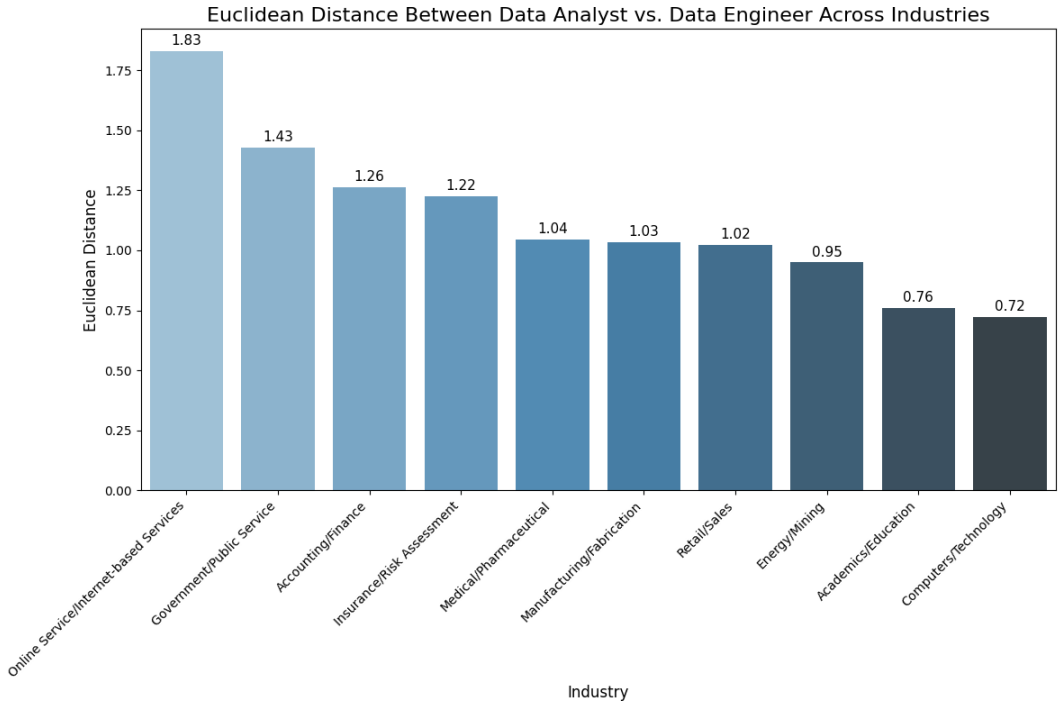


Figure 5: Euclidian distances Between Data Analysis and Data Engineer: *In the plot, each bar represents an industry. Higher distances in Online Services (1.83) and Government/Public Service (1.43) reflect greater skill differences, while lower distances in Computers/Technology (0.72) and Academic/Education (0.76) indicate more overlapping skills, facilitating smoother role transitions.*

It is important, however, to consider potential limitations of those findings due to dataset imbalance and sample size disparities. Computers & Technology sector has 1,733 observations, whereas industries like Energy/Mining and Insurance/Risk Assessment only have 176 and 199 observations, respectively. To have a complete overview of the frequency distribution of industry, check Appendix A.3. When further segmented by roles, the number of observations in these smaller sectors decreases even more, which may affect the reliability of the euclidean distance calculations and the generalizability of the findings.

These findings highlight that while a skills-first approach is generally feasible, the extent to which skills are transferable between roles are influenced by industry-specific factors. For organizations and professionals, this underscores the importance of considering industry context when evaluating skill transferability and designing workforce development programs.

4.4 Role Overlaps in Terms of Demand

The analysis conducted this far has reinforced our hypothesis that there is a significant overlap in skills among technology-related roles, supporting the feasibility of a skills-first approach in recruitment and internal skill development in the technology sector. By utilizing the Kaggle Survey dataset, we identified substantial similarities in both the relative and absolute usage of skills across various tech positions. This suggests that professionals may transition between roles more easily than job titles alone might imply, expanding the pool of qualified candidates and enhancing workforce flexibility.

However, it's important to acknowledge the limitations inherent in our initial analysis, which may affect the reliability and generalizability of the results: similarity measures based on positive value constraints, indirect skill measurements based on regular activities and tools as skill proxies, self-reporting biases, and potential overrepresentation of certain roles and industries. To address some of these limitations and validate our findings, we introduce an additional dataset collected from O*NET (Occupational Information Network). O*NET is a comprehensive database developed under the sponsorship of the U.S. Department of Labor and provides hundreds of standardized and occupation-specific descriptors on nearly 1,000 occupations covering the entire U.S. economy. The selected dataset focuses on hard skills that are in high demand across various roles, rated on a scale from 0 to 100 based on their frequency in job postings.

4.4.1 Rationale for Using the O*NET Database

The O*NET dataset serves as an effective validation source for several reasons. First, it offers objective skill measurements. Unlike the self-reported Kaggle data, O*NET's data is derived from research and expert input, making it a more reliable indicator of required hard skills. Second, O*NET data provides a demand-side perspective. While Kaggle data reflects the skills

professionals currently possess, O*NET captures the skills that employers are seeking, offering a complementary view on skill alignment. Finally, O*NET encompasses a broad range of occupations and industries, using standardized data collection methods that help address issues related to unbalanced role representation.

We mapped the roles analysed in the Kaggle dataset to their closest equivalents in O*NET as presented in Table 2:

Kaggle dataset	***O*NET***
Data Scientist	*Data Scientist*
Data Engineer	*Computer Systems Engineer/Architect*
Data Analyst	*Business Intelligence Analyst*
Software Engineer	*Software Developer*

Table 2: Role Mapping: *The table illustrates the role mapping between the two datasets, ensuring consistency in the analysis and enabling meaningful comparisons of results across different roles.*

The ML Engineer role was not directly available in O*NET; therefore, we sourced relevant skill demand data from external sources, focusing on key ML competencies pertinent to this role. After merging all datasets, the final DataFrame used for the analysis consisted of 196 features on different technological related skills.

Using the O*NET data, we recalculated similarity measures between roles employing cosine similarity and euclidean distance metrics based on the skills frequencies in job postings. This approach ensures consistency in evaluating skill overlaps across both datasets, allowing us to verify whether the patterns observed in the Kaggle data hold true for the O*NET dataset.

4.4.2 Findings From O*NET Validation

Upon analysing the O*NET dataset, we observed that the euclidean distance between roles generally appeared to be low, ranging from 0.327 to 1.21. Scores for all roles are available in Appendix A.4.2. This indicates close relationships among roles, suggesting a high degree of skill overlap.

Furthermore, examining the overlap of specific roles based on the O*NET dataset reveals both similarities and differences compared to the findings derived from the Kaggle dataset. The ML Engineer consistently shows higher euclidean distance with other roles in both datasets, reaffirming its specialized nature. Conversely, the Research Scientist, which previously exhibited low similarity with other roles in the Kaggle dataset, now shows closer distances with Data Analysts, Data Engineers, and Data Scientists in the O*NET data. Moreover, we find that only two out of six roles maintained the same top overlapping role when analysed using the O*NET data.

The observed variations can be attributed to several factors. One primary factor is the differing definitions of roles across datasets. For instance, while a Data Engineer and a Computer Systems Engineer/Architect are mapped as equivalent roles, subtle distinctions in their responsibilities and required skills can lead to variations in similarity measures. Additionally, discrepancies between the Kaggle and O*NET findings arise from the inherent nature of the datasets themselves. The Kaggle dataset reflects the skills that professionals report possessing (supply side), whereas O*NET captures the skills that employers seek (demand side). These differing perspectives naturally result in variations, as employers may prioritize different skills based on current market needs and technological advancements. Moreover, O*NET encompasses a broader range of features directly related to skills, which likely contributes to the observed differences in similarity measures.

However, these results should be interpreted with caution due to the positive-value constraint in the O*NET data. Since skill usage is represented as percentages constrained between 0 and 100, this limits the range of potential distances. Consequently, the observed euclidean distances may appear relatively low, even when meaningful differences exist.

4.5 Future Improvements & Developments

Dataset: The perfect dataset would address key limitations in current datasets from Kaggle and O*NET. It would capture not only whether a skill is possessed or required but also the proficiency levels demanded for various roles. For instance, the depth of coding expertise needed for a software engineer compared to a data analyst could greatly impact transition feasibility. Additionally, to mitigate self-reporting bias, the ideal dataset would include objective assessments of individual skill levels, such as standardized tests or certifications. Moreover, incorporating factors such as industry-specific requirements, years of experience, and non-technical attributes like soft skills and adaptability would create a more holistic understanding of role transitions.

Methodology: To improve the current analysis, the methodology for measuring shared skills can be enhanced using skill embeddings (e.g., Word2Vec or Transformers). These methods go beyond binary skill overlaps to capture nuanced relationships between skills, such as how proficiency in one skill (e.g., Python) relates to another (e.g., SQL). Textual data on job responsibilities and daily activities could also be leveraged using pre-trained embeddings to identify implicit skill similarities. Beyond cosine similarity, alternative measures like Manhattan Distance for sparse data or TF-IDF weighting to emphasize skill importance could add depth. Additionally, kernel methods (e.g., RBF) can explore non-linear relationships, offering more sophisticated similarity modelling.

Expanding the analysis could include training classifiers (e.g., Random Forest, Gradient Boosting) to predict transition feasibility based on a richer feature set, as well as recommendation systems. For example, collaborative filtering could recommend roles based on others' successful transitions, while content-based filtering could leverage skill overlaps.

Lastly, advanced gap analysis using ML can identify skills most associated to successful transitions, offering actionable insights for upskilling.

Possible Use Cases: A skills-first approach opens exciting possibilities beyond role transitions. One interesting application is assessing alignment between skill demand and supply, helping organizations identify areas where workforce skills lag market needs. Another use case is building a classification model to match roles with unemployed individuals based on their existing skill sets and potential for targeted upskilling. Such applications would not only enhance workforce planning but also promote inclusive growth by enabling more people to transition into high-demand roles.

4.6 Main Findings

Our analysis strengthens our hypothesis that a skills-first approach to recruitment and workforce development in the tech sector is both feasible and advantageous. Using a dual-dataset methodology to provide complementary perspectives, the findings show a consistent overlap in skills in both the supply side (Kaggle) and the demand side (O*NET). This alignment highlights the potential of a skills-first strategy to expand talent pools and facilitate smoother career transitions.

However, both similarity calculations were limited by bounded positive values, potentially resulting in inflated similarity scores. To address these challenges, the skills distribution analysis looked deeper into actual skill usage to better understand the factors behind these high similarity scores. While the analysis revealed consistency in skill usage across many roles, it also identified positions with more specialized skill sets, such as ML Engineer and Research Scientist. This suggests that although transitions between some roles can be relatively easy, other positions still demand targeted upskilling. Moreover, our findings highlighted the importance of considering industry-specific nuances. Industries such as Government & Public

Sector and Insurance & Risk Assessment showed higher euclidean distances between role pairs, suggesting more specialised skill requirements. By acknowledging these nuances, organizations can more effectively utilize a skills-first approach, supporting more dynamic career paths, optimizing talent allocation, and ultimately driving greater agility in workforce planning.

5 Model Development: Skills-First Résumé Matching

5.1 Motivation

Given the potential benefits of the skills-first approach as outlined above, we set out to add value to the field by developing **Skillmatch.ai**, an all-in-one AI-driven ecosystem for recruiting and skill management. As indicated in the literature review, résumé screening has traditionally been a time-consuming task, constraining the resources of HR departments. This has prompted the introduction of many résumé screening tools, both keyword-based approaches and more complex approaches leveraging modern NLP techniques and LLMs. To our knowledge though, there are few such tools combining these techniques with the skills-first approach.

Our baseline model focuses on matching candidates to job descriptions based on their skills. To enhance this core functionality, we introduced additional features aimed at reducing the workload on HR departments. These include leveraging LLMs to generate tailored interview questions for candidates advancing to the first round of interviews, as well as providing constructive feedback to those who do not proceed, helping them improve their skills for future applications. Additionally, we added functionality to help users identify roles that align with their current skill set and explore potential salary changes when transitioning to these roles.

Skillmatch.ai does not only provides a robust matching algorithm for pairing job applicants with roles within a company, but also offers functionalities to enhance the entire recruitment process by addressing gaps in the HR and recruiting product space. Together, these features have the potential to transform the way businesses approach recruitment, providing a holistic, skills-first solution that creates value for both employers and job seekers. In the coming section we will focus on the development of the résumé screening matching algorithm while we will elaborate on the other functionalities and the web application in sections 8, 9 and 10.

5.2 Building the Matching Model

5.2.1 Overview

The first step in the model pipeline involves parsing and preprocessing résumés and job descriptions using libraries like PyMuPDF and pandas. Next, relevant technological skills were extracted using NLP techniques with spaCy. These skills are then converted into vector representations using an MPNET-based model for semantic analysis. Finally, cosine similarity scores are calculated to rank candidates based on how well their skills align with the ones stated in the job description. It should be noted that our work is built upon the work of Amira Dridri (2022) and her job-résumé matching model. Her project does not include an open-source license, but we received written consent from the author to borrow parts of her code for our own project. A full overview of the relevant code and the file tree can be found in Appendix B.1. Moreover, our entire codebase is provided open source at: https://github.com/victorbjorsvik/workproject_matching_algo

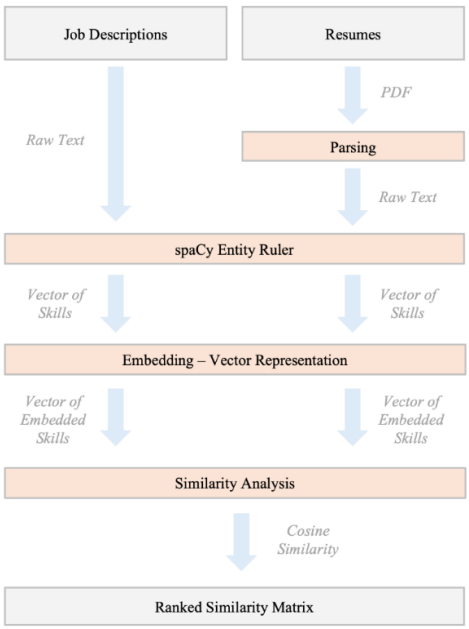


Figure 6: Matching Model Pipeline: *The pipeline starts with two inputs, résumés in PDF format and job descriptions in raw text. First, we extract raw text from pdf formatted résumés. Next, we use the spaCy Entity Ruler to extract skills from the two inputs. We then embed the skills using a semantic embedding model (in our case MPNET) before we conduct a similarity analysis with cosine similarity.*

5.2.2 Note on Validation Data

Ideally, we would have a labelled dataset from the real world, annotated by a recruiter employing the skills-first approach during the recruitment process, to validate the efficacy of our model. However, obtaining such a dataset proved challenging due to privacy and regulatory concerns. Additionally, the nature of the labels themselves poses unique challenges in our context. For instance, should the labels be binary (e.g., "got job" or "did not get job") or continuous (e.g., productivity after employment, suitability scores)? Should they reflect outcomes like whether the applicant received an interview, was hired, or their performance post-employment? These ambiguities make it difficult to establish a clear, standardized dataset for validation purposes. We set aside two weeks to search for such a dataset, even compromising on the assumption that the recruiter had to employ the skills-first approach, but we were unable to procure one. To decide on the main components of our model, we relied on a small synthetic dataset, intuition about the relevant concepts, and insights from existing literature.

We acknowledge the limitations of this approach and in section 6 and 7 of this paper we conduct a full-scale validation of our model using a much larger synthetic validation dataset to evaluate its performance and identify areas for improvement. Furthermore, once the model is deployed, we plan to begin collecting real-world data from users. This will enable us to iteratively refine and improve the model by using feedback and outcomes from actual recruitment scenarios. As such, we will not conduct any comprehensive evaluation of the model in this section of the paper, but rather provide insights into the model structure and its main components.

5.2.3 PDF Parsing and Preprocessing

The initial phase of our model pipeline involves parsing and preprocessing the résumés. This is achieved by extracting text from PDF files using the PyMuPDF library and processing the data with Pandas. This preprocessing step is crucial as it converts the data from the PDF files into

raw text, enabling further analysis and processing. There are multiple Python libraries for parsing PDFs. We started out with one of the most popular libraries, pypdf2, but after extensive testing and research we found that the PyMuPDF library achieved significantly faster parsing times. This is in line with the findings of Thoma et al. (2023) in their python PDF parsing benchmarks. The code for this step can be found in Appendix B.2.

5.2.4 Skills Extraction

In the skill extraction step, the goal is to identify and extract the relevant skills from the résumés and job descriptions. The inputs for this step are the data frames derived from the PDF parsing, containing the raw text from the résumés and the job descriptions. We utilize the spaCy library along with a custom pattern file to extract the relevant skills. SpacCy is an open-source library for advanced NLP in Python. It offers powerful tools for processing and analysing large volumes of text, including entity recognition, part-of-speech tagging, and dependency parsing (Honnibal and Montani 2017). The custom pattern file is a structured dictionary containing predefined patterns for recognizing specific skills in the technology sector. This file is loaded into spaCy's EntityRuler, which matches these patterns against the text parsed from the résumé. The pattern file can be found in the "*Resources/data/skills.jsonl"* directory of our codebase and was borrowed directly from Dridri (2022) with some minor tweaks and additions. The output of the skill extraction step are arrays of relevant skills – both for each application and for the job description.

As an alternative approach, we considered ontology-based methods, which involve mapping extracted entities to external knowledge bases like ESCO and O*NET to standardize terms and ensure consistency (Shakya and Paudel 2019). However, we did not choose this approach because ontology-based methods add complexity and require advanced mapping logic, whereas pattern-based extraction offers a direct and flexible way to extract and structure relevant data.

Moreover, our structured pattern file is tailored towards the technology sector and therefore holds a more extensive and relevant basket of skills compared to the most popular ontologies which are more general in nature. The entity ruler approach means that we are missing out on the inherent hierarchies and connections present in the ontologies, but since we are using MPNET embeddings some of these connections will likely be picked up in the semantic embedding space of the skill vectors. As an additional approach we tried creating our own custom pattern file based on a subset of technical skills from O*NET. We called the OpenAI API in batches of 50 skills at a time to convert the list of skills into a *jsonl* pattern file like the one used for our original model. This process extended our list of technical skills from ~2750 to 8767. However, the results on our small validation set were not promising, likely because the LLM generated only one distinct pattern for each unique skill. With additional work and curation this could have been a valid approach, giving us a much richer vocabulary of skills, but given time constraints and other prioritizations we decided to go for the initial *skills.jsonl* pattern file. Code snippets for the extraction steps of our pipeline are presented in Appendix B.3.1 and B.3.2.

5.2.5 Embedding and Semantic Representation using MPNET

Computers and ML models inherently operate on numerical data; thus, converting text into numerical vectors is a critical step in our pipeline. The simplest possible approach in our context would have been to create a strict union between the array of skills from the job descriptions and the résumés. Since the spaCy entity ruler would return each skill in the same format this would have been a viable approach. On the other hand, this would only allow us to capture the strict overlap of skills and not account for the fact that certain skills are very similar to each other. As such, we will consider some alternative NLP techniques.

Traditional Approaches (BoW, TF-IDF): Traditional approaches have been to use one-hot-encoding for every single word in the training corpus, approaches like Bag-of-Words (BoW) or other term frequency schemas like Term frequency-inverse document frequency (TF-IDF) (Han, Kamber and Pei 2012). While straightforward and computationally efficient on short documents, these approaches would have several limitations in our application. For one, these approaches can yield sparse matrices and lead to high dimensionality if the input is long. Essentially, we would have an n_skills $(\sim 2750) \times$ $n_applicants$ and an n_skills $(\sim 2750) \times$ $n_jobdescriptions$ where a good majority (> ~2730) of the columns would be encoded to 0 for the one-hot-encoding approach since a résumé or job description rarely contain more than 20 unique skills. This would have been computationally inefficient and would severely affect the cosine similarity metrics. Second, these approaches treat the skills as independent entities without any inherent semantic meaning. Lastly, they lack contextual awareness, meaning that these approaches will not capture elements like polysemy (words can have different meanings) and synonymy (different words can have the same meaning).

Non-Contextual Embeddings (Word2Vec, GloVe): Another category of methods – non-contextual word embeddings – includes models like Word2Vec and GloVe. They improve upon the traditional methods by capturing semantic relationships between words based on their co-occurrence in large text documents. Word2Vec estimates the vector representations of the words through a shallow neural net where words appearing in similar contexts are positioned closely in the vector space, effectively capturing semantic similarities (Mikolov, et al. 2013, a, b). GloVe (Global Vectors for Word Representation) extends this by analysing how often words appear together across a large collection of text (word co-occurrence). This approach balances two key strengths: capturing the overall relationships between words (global patterns, such as "king" and "queen" being related) and understanding how words are used in specific contexts (local patterns, like "king" often appearing near "castle"). By combining these elements, GloVe

produces embeddings that effectively represent both broad semantic relationships and finer-grained contextual meanings (Pennington, et al. 2014).

Using these word-embedding models could improve our similarity analysis because words like “C” and “C++” would likely be positioned closely in the embedding space and thus yield a higher cosine similarity, assuming that both “words” are present in our embedding vocabulary. On the other hand, these methods still generate static embeddings, meaning each word has a single representation regardless of context. This will likely pose problems for our model because words like "Python" could refer to a reptile rather than the programming language. Although "Python" would be embedded equally for the job posting and the résumé, and as such would yield a high similarity for this skill pair, it would not, however, yield high similarity with another programming language like JavaScript or Go.

An alternative would have been to use embeddings tailored to the technology sector. Then, we would have both the semantic representation in the embedding space and the semantic meaning would be specific to our domain. Such embeddings do exist in the literature. For instance, Van-Duyet et al. (2017) trained a Skill2Vec model to capture the embeddings of skills from a large corpus of job descriptions. We tried using these embeddings for our own model, but one issue is that their skill vocabulary does not match our own. This would pose problems as our spaCy entity ruler could return skills which cannot be embedded with the Skill2Vec embeddings. An alternative approach would be to train our own embeddings from scratch, but finding enough data to capture meaningful co-occurrence between every single skill in our skill dictionary proved difficult.

Contextual Embeddings (ELMo, BERT, MPNET): To overcome these challenges, we used another category of embeddings – contextualized embeddings. The embeddings from this category are often derived from Large Language Models (LLMs) trained on large text corpora

and they work on a character - or token-level rather than word level. The key difference from previous methods is that contextualized embeddings allow tokens in the input sequence to interact, enabling the embeddings to capture the semantic meaning of each word within the context of the entire sequence. This is particularly advantageous for our application because it 1) **captures context**: By understanding the context, these models can distinguish between different meanings of the same word, ensuring that "Python" in the context of programming is semantically closer to "Java" or "C++" rather than to "snake" or "reptile." 2) **Can handle synonymy and polysemy**: Contextual embeddings allow for better handling of synonyms and words with multiple meanings, improving the accuracy of similarity calculations. 3) **Bigger vocabulary:** since the model operates on a token-level and the tokens are derived from byte pair encoding, we are at less risk of not being out of vocabulary when we try to embed our skills. And 4) **leverages pre-trained knowledge**: several of these models have been trained on large parts of the internet. This enables us to utilize a vast amount of linguistic knowledge without the need for extensive domain-specific training data.

The contextual embedding models became popular with the introduction of ELMo (Peters, et al. 2018) – an embedding model based on the LSTM architecture. The advent of the transformer architecture with the famous “Attention is All You Need” paper (Vaswani, et al. 2017) gave rise to several other contextual embedding models though, and in 2019 the team at google modified the original transformer architecture by stacking several encoder blocks to let the tokens communicate bi-directionally in the sequence. This gave rise to the BERT family of models (Devlin, et al. 2019). We are employing the “all-mpnet-base-v2” from the Hugging Face Sentence Transformers library. This model builds further upon the BERT framework by leveraging insights from XLNet (Yang, et al. 2019). This model beats the original BERT models in several important NLP benchmarks like General Language Understanding Evaluation

(GLUE) and Question Answering (SQuAD) (Song, et al. 2020). Moreover, it is also the most downloaded model from the Sentence Transformers library (HuggingFace 2024).

5.2.6 Similarity Analysis

After embedding the arrays of skills into the semantic space using the MPNET model, we proceed to calculate the similarity between the résumé skills vectors and the job description skills vectors. This step is crucial for determining how well a candidate's skills align with the requirements of a job posting. Several key considerations influenced our approach to similarity analysis:

Choice of Encoder Architecture: Bi-encoders vs. Cross-encoders: One of the initial decisions was whether to use a bi-encoder or a cross-encoder architecture. Bi-encoders independently encode each input (in our case, the list of skills from résumés and job descriptions) into fixed-size embeddings. These embeddings can then be compared using similarity measures like cosine similarity. Cross-encoders, on the other hand, take pairs of inputs simultaneously and output a similarity score directly, potentially capturing richer interactions between the inputs.

While cross-encoders have demonstrated superior performance on certain tasks (Reimers and Gurevych 2019), they are significantly more computationally intensive. Given the need to efficiently compare numerous résumés with various job descriptions, we opted for the bi-encoder approach. This choice strikes a balance between computational efficiency and performance, making it ideal for large-scale applications. Additionally, the bi-encoder yielded better results on our small-scale validation set.

Skill Representation: Skill-by-skill vs. Basket of Skills: Another critical consideration was how to represent the skills during embedding. Dridri (2022) used a strict equality check with a cosine similarity threshold of 0.4, comparing skills on a skill-by-skill basis. However, this

method does not leverage the contextual capabilities of our embedding model and is computationally inefficient due to the large number of pairwise comparisons required. Moreover, if the spacy EntityRuler is only able to extract a few skills, this method might have a propensity towards high scores because several applicants might possess all the required skills, and this will give them a perfect score of 1 through the strict equality check.

We chose to embed the entire list of skills as a "bag of skills," treating them as a single sequence. This approach allows the model to capture the collective semantic meaning and interactions between skills. However, concatenating skills into a single sequence can lead to tokenization issues, where individual skills may not be distinctly recognized, potentially affecting similarity calculations. To mitigate this, we introduced the special end of sentence token from the Hugging Face Tokenizer between each skill before embedding. This practice helps the model distinguish between individual skills, likely because the model is better able to separate each skill in the tokenization process, while still considering the context provided by the entire list. This method enhanced the model's ability to capture nuanced relationships between the candidate's skills and the job requirements. See Appendix B.4 for the results after the implementation.

Similarity Metric: Cosine similarity: For the similarity calculation itself, we employed cosine similarity, as introduced in section 4. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, effectively capturing their directional alignment. It is widely used in NLP and information retrieval because it is unaffected by the magnitude of the vectors, focusing solely on their orientation (Han, Kamber and Pei 2012).

Cosine similarity is particularly appropriate for high-dimensional and sparse data, common in text representations (Han, Kamber and Pei 2012). It provides an efficient and interpretable measure of similarity that aligns well with our goal of matching résumés to job descriptions based on skill sets. By using cosine similarity, we can quantify how closely a candidate's

combined skill vector aligns with that of a job posting, facilitating a ranked list of candidates based on their suitability. Since the goal of our base model is to select a pool of candidates to proceed to the first round of interviews, we select the top K candidates based on their similarity scores. Code snippets for our embedding steps and similarity calculations can be found in Appendix B.5.

5.2.7 Code Optimization

After implementing all components of our model pipeline, we were able to run the model with a job description and a collection of résumés in PDF format as input. The output is a table listing applicants along with their respective cosine similarity scores relative to the job description. However, we observed that the model's performance was slow, especially when processing a large number of candidates. This posed a challenge for practical applications where timely results are crucial. To enhance the model's efficiency and scalability, we undertook several code optimization steps, detailed below. We measured the improvements on the mean time of 3 iterations for a batch of 6 applications and 1 job description.

1) Initialization of spaCy NLP Instance in the Class Constructor: In the initial implementation, a new instance of the spaCy NLP model was being loaded every time the skills extraction function was called. This redundant loading process incurred substantial overhead, especially when processing multiple résumés. To optimize this, we initialized the spaCy NLP instance once in the class constructor. By doing so, the NLP model is loaded only once when the class is instantiated, and the same instance is reused for all subsequent calls to the skills extraction function. This change significantly reduced the cumulative loading time of the NLP model, thereby improving processing time by ~ 18.27%.

2) Utilization of the 'apply' Method Instead of Row Iteration: Previously, we were iterating over the rows of pandas DataFrames using explicit loops to process each résumé and job

description. This approach is inherently slow in Python due to interpreter overhead and lack of vectorization. To optimize, we replaced explicit row iteration with the pandas apply method, which applies a function along an axis of the DataFrame. The apply method is more efficient because it leverages underlying optimizations in pandas and can process data in a vectorized manner. This step did not significantly improve processing times for our small batch – but we would expect to see significant reductions with larger batches.

3) Vectorization of Similarity Calculations: The similarity calculations between the embedded skill vectors of résumés and job descriptions were initially performed in a pairwise manner using explicit loops. This approach was not scalable and resulted in slow performance as the number of candidates increased. To overcome this, we optimized the process by vectorizing the similarity calculations, computing the cosine similarity between all roles and the job description simultaneously, rather than handling job descriptions and roles iteratively in a loop. This leveraged optimized linear algebra operations provided by libraries such as NumPy, enabling batch processing of similarity scores. Vectorization significantly reduced computation time by ~ 15.41% and allowed the model to handle larger datasets more efficiently.

4) Optimization of PDF Text Extraction: We initially used the PyPDF2 library for extracting text from PDF files. While functional, PyPDF2 demonstrated slow performance, especially with large documents or a high volume of files. Based on our research and testing, we switched to the PyMuPDF library, which offers faster text extraction capabilities. This change aligned with the findings of Thoma et al. (2023), who benchmarked various Python PDF parsing libraries and found PyMuPDF to be significantly faster. By optimizing the text extraction step, we reduced the time required to preprocess the résumés by ~18.90%.

5) Parallelization of MPNET Embeddings: Generating embeddings using the MPNET model is computationally intensive, particularly when dealing with a large number of résumés. To

optimize this step, we enabled parallelization by utilizing half of the available CPU cores during the embedding process. This parallel processing was facilitated by libraries that support multi-threading or multi-processing in Python, such as the *joblib* library. By distributing the workload across multiple cores, we were able to generate embeddings in parallel, significantly reducing the total computation time by ~25.66% in our small test batch.

Summary: Through these optimization steps, we significantly improved the performance and scalability of our model pipeline. By reducing redundant operations, leveraging vectorized computations, optimizing data extraction, and utilizing parallel processing, we were able to achieve faster execution times without compromising the accuracy of the model. These improvements make the model more practical for real-world applications, where efficiency and scalability are critical factors. In total, we managed to reduce the total processing time for our small test batch by ~55.03%.

5.3 Limitations and Future Development

While the model shows promise as a skills-first approach to résumé matching, there are several limitations to consider. First, the absence of a domain-specific ontology. While the model leverages a custom pattern file for skill extraction, this approach is constrained by the predefined patterns and may fail to capture emerging skills, alternative terminologies, or nuanced relationships between skills. Ontology-based methods, which offer structured hierarchies and standardized terms, could enhance the model's semantic understanding. However, implementing such methods requires advanced mapping logic and extensive domain-specific data. Exploring hybrid approaches that integrate ontologies with contextual embeddings presents a promising avenue for improvement.

Scalability remains another area of concern, particularly when processing large datasets or operating in real-time applications. Although optimizations such as vectorized computations

and parallel processing have improved efficiency, the computational demands of embedding generation and similarity analysis are still significant. Future work will focus on optimizing resource allocation and exploring lightweight embedding techniques to enhance scalability without compromising performance.

The reliance on predefined skill patterns introduces potential biases, as these patterns may not comprehensively represent all relevant skills or account for domain evolution. Additionally, while the MPNET embeddings provide contextualized representations, their lack of fine-tuning for the recruitment domain may limit the model's ability to fully capture the intricacies of skills-specific matching. Fine-tuning these embeddings on domain-specific corpora or leveraging sector-specific pretrained models could significantly enhance performance.

The absence of robust evaluation metrics represents another limitation. Without access to real-world labelled datasets, the model's efficacy cannot be fully assessed against industry-standard metrics such as precision, recall, or F1-Score. This limitation underscores the importance of securing labelled datasets and establishing clear benchmarks for performance evaluation. Future efforts will prioritize the collection of real-world labelled datasets to enable robust evaluation and iterative refinement of the model.

The model also faces challenges in handling multi-skill roles, where balancing the relative importance of diverse skill requirements is crucial. Current methods may bias the model toward candidates with narrowly overlapping skill sets, potentially overlooking candidates with broader but equally relevant skill profiles. Developing mechanisms to weigh skills dynamically based on their importance within specific roles could address this issue.

Finally, the interpretability of recommendations poses a challenge. While the model provides similarity scores for candidate-job matches, it does not offer transparent explanations for these scores. This lack of interpretability may limit its adoption by recruiters who require insights

into the rationale behind AI-driven decisions. Incorporating explainability frameworks into the model, such as feature attribution methods, will be a critical step in addressing this limitation.

Moving forward, we aim to address these limitations through several initiatives. The immediate priority is to validate the model using the synthetic data generated under section 6. Post-deployment, iterative refinement will be driven by continuous data collection and analysis. Expanding the skill dictionary and incorporating ontologies will enhance the model's ability to adapt to evolving domains. Moreover, exploring unsupervised or semi-supervised learning techniques will help the model better adapt to new datasets with minimal labelling effort. Future research will also focus on integrating explainable AI methodologies to improve transparency and user trust.

While the current iteration of the model lays a solid foundation for skills-first recruitment, these planned improvements aim to ensure that it evolves into a robust, scalable, and adaptable tool capable of meeting the diverse needs of modern recruitment processes.

6. Generating Synthetic Data to Overcome Data Accessibility Challenges for Model Testing and Evaluation

This chapter outlines a structured approach to address data accessibility challenges by generating synthetic data for testing and optimizing our proposed skills-first matching model pipeline. Evaluation is highlighted as critical for refining the model and guiding improvements. Due to privacy and resource constraints, synthetic data was used to approximate an ideal dataset, employing processes such as collecting real-world job descriptions, iterative prompt optimization with OpenAI's GPT-4o-Mini, and dataset refinement. While this methodology aims to facilitate robust evaluation and scalability, limitations, including potential biases and reduced real-world granularity, emphasize the need for cautious interpretation and ongoing refinement.

6.1 The Critical Role of Evaluation in Model Development

Evaluation is critical for assessing the effectiveness of our proposed model and guiding its refinement. Within the adapted CRISP-DM framework, it ensures alignment with technical and business objectives, identifies weaknesses, and supports decisions to deploy, iterate, or reframe the modelling approach (Wirth and Hipp 2000). This iterative process safeguards against performance discrepancies in real-world scenarios and provides actionable insights for optimizing algorithms (Hossain 2024; Raschka 2018).

Our proposed model pipeline relies on synthetic data, domain knowledge, and literature insights to define its components. Given the abundance of state-of-the-art methods, benchmarking selected components against alternatives is essential. Effective evaluation involves labeled data serving as ground truth, enabling calculation of metrics like precision, recall, and F1-Score (see explanation of metrics in Appendix D.1), which highlight model strengths and areas for improvement (Freire and de Castro 2021).

6.2 Problem Statement

Acquiring high-quality labelled data for model evaluation posed significant challenges. Data protection regulations, including GDPR, classify résumés as personal data, restricting companies from providing them without explicit consent (Council 2024; Voigt and Bussche 2017). Despite efforts to access data through well-regarded data providers, such as Nova SBE's Data Science Knowledge Center and Lightcast.io, as well as public repositories, no suitable datasets were identified.

Data accessibility challenges also stem from the resource intensity of creating labelled datasets, which demands both financial resources and expertise – particularly in the case of skills-first hiring, where tailored labelling efforts are required (Ratner, et al. 2017; He and Garcia 2009). This issue is reinforced by missing prioritization and frameworks often faced by companies trying to adopt skills-first practices, as mentioned in Section 3.2.4. Without those, efforts such as customized data labelling are unlikely to receive the necessary organizational resources or priority.

Even if datasets with the required format were available, their alignment with a skills-first approach can pose another challenge. While companies generally emphasize skills in hiring, decisions may still be influenced by degree titles and personal fit (Section 3.1.1.). These factors are outside of the scope of our proposed skills-first matching approach, which focuses solely on skills. Consequently, real-world hiring data may introduce noise, as hiring outcomes can reflect traditional hiring criteria rather than skills-based decisions, making it difficult to reliably evaluate a model designed to focus on skill alignment exclusively.

These limitations underscore the need for synthetic data as an alternative solution to benchmark and optimize the proposed model pipeline effectively. Synthetic data can address gaps in availability and alignment, enabling a controlled evaluation environment.

6.3 Ideal Data Solution for Testing and Evaluation

To ensure a reliable and skill-focused model evaluation process, we propose an ideal dataset structure enabling precise benchmarking and optimization of the model pipeline. The essential components of the dataset, methodology for its creation, and implications for evaluation are outlined below (Figure 7).

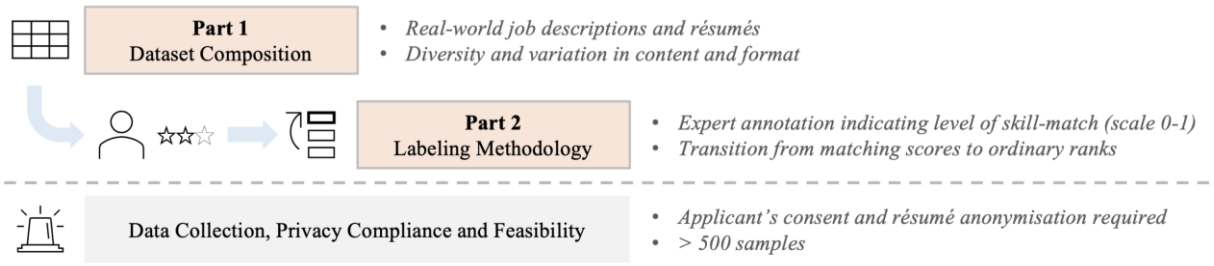


Figure 7: Methodology for the creation of the ideal dataset for model testing and evaluation: *The visualization outlines the steps required to obtain the necessary dataset for model testing and evaluation, from inputs to the labelling process to important privacy and feasibility considerations.*

Dataset Composition: The dataset should include real-world job descriptions and résumés as inputs, with expert-provided labels indicating skill alignment as the ground truth. Job descriptions should represent a diverse range of roles in the technology sector, reflecting typical ambiguities and variations. Résumés should capture the diversity in structure, detail, and presentation of skills, whether explicitly listed or embedded in text, to test the model's ability to handle imperfect formats effectively.

Labelling Methodology: Expert annotations should indicate the degree of skill alignment between job descriptions and résumés. According to Freire and de Castro (2021), expert validation is an established evaluation method for recommendation systems. Domain experts with relevant knowledge would review each résumé for a specific job description, assessing the skill alignment in terms of explicit and related skill matches and assigning a score on a scale from 0 (no fit) to 1 (perfect fit), providing a precise measurement of alignment. These scores enable precise evaluation of skill alignment and facilitate the translation of raw scores into ordinal ranks, which the model can replicate. This approach is necessary because the

computation of similarity scores is not directly observable using pretrained models for embedding, making precise score replication infeasible. Alternatively, binary classification labels (“good fit” or “bad fit”) offer a simpler yet effective method for evaluation, enabling key metrics like precision, recall, and F1-score (Appendix D.1). Binary labelling reduces complexity and manual effort, making it a practical solution in scenarios with resource constraints (Liu, et al. 2024; Speranskaya, Schmitt and Roth 2021; Otani, et al. 2020).

Data Collection and Privacy Compliance: The data collection process must adhere to ethical and legal standards, particularly in line with data privacy regulations such as GDPR. Each candidate whose résumé is included must provide explicit consent for their data to be used in the process. This ensures compliance with privacy laws and guarantees transparency in how personal information is handled. All résumés must be anonymized and used solely for the purposes of this research in order to comply with ethical and legal standards.

Feasibility and Dataset Scale: Given the resource-intensive nature of data collection and expert labelling, we propose that the ideal dataset should comprise 100 job descriptions, each paired with five to six résumés depending on the type of labelling used, resulting in a total of 500 – 600 labelled job descriptions-résumé pairs. According to Rajput, Wand, and Chen (2023) this size is sufficient and strikes a balance between the manual effort required for labelling and the dataset’s utility for benchmarking.

Implications for Model Evaluation: The proposed dataset design addresses key challenges in evaluating skills-based hiring models and allows for robust benchmarking. The inclusion of real-world job descriptions and résumés ensures realistic evaluation conditions and tests the model’s ability to generalize effectively. Labels generated by experts provide an accurate measure of skill alignment, focusing solely on relevant skills and avoiding biases introduced by traditional hiring criteria. Finally, the proposed design supports both detailed and simplified

labelling, offering flexibility to adapt to varying project resources while maintaining evaluation validity.

6.4 Limitations of the Proposed Data Collection Approach

While the proposed ideal dataset offers a robust framework for evaluating skills-based hiring models, there are several potential limitations and feasibility challenges.

Reliability and evaluation limitations: The proposed scoring and labelling system depends on subjective assessments by domain experts. Despite best efforts, inconsistencies and potential biases in expert judgments could impact the validity and reliability of the ground truth. While ordinal labelling simplifies the evaluation process, it may lack the granularity needed to assess subtle differences in candidate-job matches. This limitation could reduce the model's ability to make fine-tuned distinctions in high-stakes scenarios.

Dataset scale and generalizability: Although 500 – 600 labelled pairs are suggested as sufficient (Rajput, Wand and Chen 2023); this scale may still be too limited to fully capture the diversity and complexity of real-world hiring scenarios. Variations across industries, roles, and regions may require a substantially larger dataset to ensure the model's generalizability and reliability. Although the dataset size is considered manageable, scaling this approach to evaluate larger, more complex models may necessitate exponentially more data. This creates a scalability issue that could hinder future iterations of model testing and development.

Resource challenges: The reliance on expert annotations for skill alignment introduces significant resource demands, including the recruitment of domain experts and the time required for thorough reviews. This complexity could pose barriers for organizations with limited budgets or access to skilled professionals. Additionally, obtaining explicit consent from candidates for the use of résumés, even anonymized, presents logistical and legal hurdles. Achieving GDPR compliance in practice involves thorough processes that may delay data collection and increase costs.

By acknowledging and addressing these limitations, future research and implementations can refine the proposed dataset structure and enhance its feasibility for widespread application.

6.5 Synthetic Data Approach

The challenges of limited data accessibility and the misalignment of existing datasets with our model requirements highlight the need for exploring innovative approaches, such as synthetic data generation, to ensure a robust and meaningful evaluation process. While we have outlined how an ideal dataset could be structured and obtained, creating such a dataset requires significant resources, including expert collaboration and organizational support. Within the scope of this project, we lacked the capacity to partner with a company and produce an expertly annotated dataset. Consequently, we turned to synthetic data generation as an alternative method to approximate this ideal dataset, enabling model testing and evaluation despite resource constraints.

Synthetic data refers to data generated by a model, often created to replace or complement real data in tasks where access to actual data is limited or infeasible. It serves as a versatile tool in data-driven research and development, enabling prototyping, verification, and validation of model pipelines (Jordon, et al. 2022). A key purpose of synthetic data is to substitute real data to address privacy concerns, such as those related to GDPR (e.g., using résumés for labelling purposes). By bypassing these restrictions, synthetic data significantly reduces time and effort in the development lifecycle while mitigating disclosure risks (Rubin 1987, 1993; Little 1993; Jordon, et al. 2022). In addition, synthetic data offers a practical solution for creating consistent and controlled datasets. These datasets enable robust evaluation and benchmarking of ML models, ensuring reliability across various perspectives, especially in iterative development processes (Liu, et al. 2024). Advancements in large-scale language models (LLMs) further underscore the value of synthetic data. In many cases, synthetic judgments generated by LLMs match or exceed the quality of human annotations, providing a highly efficient and cost-

effective alternative to manual evaluation (Gilardi, Alizadeh and Kubli 2023). Finally, in the field of skill-based candidate-job matching, synthetic data has proven transformative. Frameworks like JOBSKAPE employ synthetic job postings to train classification and similarity models, reducing dependence on time-consuming and costly human annotations while improving scalability and accuracy (Magron, et al. 2024).

Despite its advantages, synthetic data comes with limitations that must be acknowledged. First, synthetic data may fail to fully capture the complexity, and nuanced patterns present in real-world data. For instance, while the ideal data solution outlined above relies on manual, expert-driven evaluation of candidate-job skill alignment, synthetic data may lack this granularity and not fully replicate expert competencies. Additionally, during the generation process the model used can introduce biases which can be propagated into downstream applications. This is particularly critical in sensitive areas like recruitment, where bias in skill matching could reinforce prejudices and skew outcomes (Hao, et al. 2024).

Understanding these limitations is essential to ensure synthetic data is used effectively. Therefore, the strategies from literature to address these challenges will be integral to the design of our own data generation pipeline.

6.6 Synthetic Data Generation Process

This section outlines the structured approach taken to generate synthetic data for testing and evaluating our skills-first matching model pipeline. The process is divided into three key phases (Figure 8): First, real-world job descriptions were collected to form the basis for generating synthetic data. Second, iterative experimentation with OpenAI's API was conducted to derive suitable prompting techniques for résumé generation, testing variations in prompt structure and labelling strategies using manual reviews and baseline model performance. Finally, the refined prompting approach was applied to a larger dataset of job descriptions, enabling the selection of the final synthetic dataset for testing and evaluation based on its impact on baseline model

performance. This stepwise approach ensured that the synthetic data was both realistic and optimized for testing and improving the model's components.

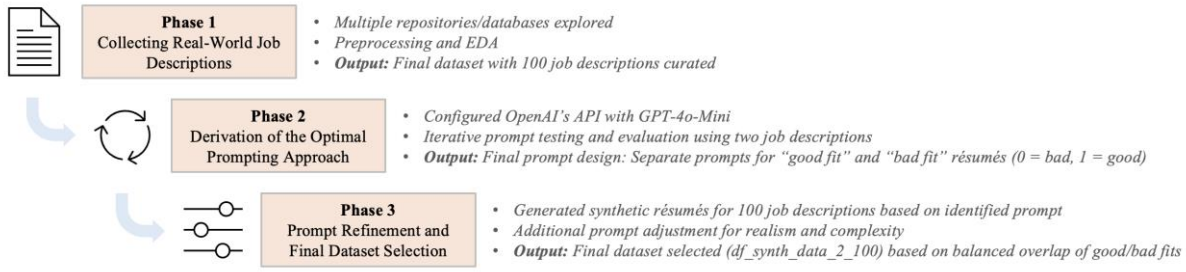


Figure 8: Synthetic Data Generation Process for Model Testing and Evaluation: *The visualization outlines the three-phase process for generating synthetic data: collecting real-world job descriptions, deriving the optimal prompt for résumé generation using OpenAI's API, and selecting the final dataset.*

Phase 1 – Collecting real-world Job Descriptions: The first phase involved sourcing and preparing 100 diverse, recent job descriptions from the tech sector. Initial efforts to access datasets from sources like Lightcast were unsuccessful due to non-responsiveness, while open-source options such as Kaggle and Hugging Face contained outdated or undated job postings. Ultimately, a GitHub dataset featuring LinkedIn job posts from 2021 was selected as the most suitable option. The dataset underwent preprocessing and EDA to ensure diversity, sufficient text length, and a representative number of skills (Appendix C.1). From 8,261 rows and 18 columns, a subset of 2,000 rows was analysed, leading to the final selection of 100 job descriptions that aligned with the evaluation goals (Appendix C.2).

Phase 2 – API setup and iterative Derivation of the optimal Prompting Approach: The second phase involved configuring the OpenAI GPT-4o-Mini model to generate synthetic résumés from job descriptions. This model, optimized for natural language generation, was selected for its ability to create human-like outputs tailored to specific job descriptions (OpenAI 2024; Gilardi, Alizadeh and Kubli 2023). System and user messages, essential components of the prompt, were utilized to interact with the AI model. The system message sets the context and guiding principles for the AI's behaviour, such as focusing solely on skill alignment, while the user message provides specific instructions for generating résumés based on the given job descriptions (OpenAI 2024). Using two manually selected job descriptions, multiple iterations

of prompting were tested to refine the approach, evaluating variations in input structure, labelling, and outputs against the baseline model. The iterative process with varying configurations is shown in Table 3.

Iteration	***Approach***	***Label***	***Outcome/Insight***
1	*5 résumés per job description generated in a single prompt.*	*Similarity score (0–1)*	*Recreating similarity scores was infeasible due to the "black box" nature of embedding models. Approach deemed unsuitable.*
2	*5 résumés per job description generated in a single prompt.*	*Ordinal ranking (1–5)*	*Generated résumés were uniformly high-quality, with similarity scores clustering between 0.76 and 0.86, resulting in limited differentiation.*
3	*Added explanations to the prompt to understand the reasoning behind rankings.*	*Ordinal ranking (1–5)*	*Differences among résumés were primarily in detail and specificity, not skill alignment, limiting the ability to distinguish meaningful variations.*
4	*3 good fit and 3 bad fit résumés per job description generated within a single prompt.*	*Binary classif. (0 = bad, 1 = good)*	*Insufficient separation in skill alignment between good and bad fit résumés due to generating both within a single prompt.*
5	*3 good fit and 3 bad fit résumés per job description generated using 2 separate prompts.*	*Binary Classif. (0 = bad, 1 = good)*	*Achieved the most promising results, with a clearer distinction between good and bad fit résumés, making this approach suitable for evaluation purposes.*

Table 3: Summary of the iterative Derivation of the Optimal Prompting Approach: The table outlines the iterative refinement process for generating synthetic résumés using OpenAI GPT-4o-Mini, detailing approaches, labelling methods, and key insights from each iteration to optimize the quality and differentiation of the outputs.

The optimal strategy involved generating three good fit and three bad fit résumés per job description using separate prompts. This method ensured a clear distinction between categories, supported by binary labels (0 for bad fit, 1 for good fit). Binary labelling aligns well with the evaluation's objective and is grounded in several key considerations and supported by literature as mentioned earlier (Section 6.3). The design of the prompting approach closely aligns with the requirements of the ideal dataset as outlined earlier, particularly regarding variability, diversity, and skill-based focus in the résumés. The system message and user message in the prompt explicitly directs the AI to create résumés tailored solely to the provided job description based on skill alignment, ensuring the generated résumés reflect the skill-centric focus emphasized for the ideal dataset (Appendix C.3). Additionally, the variability requirement is

addressed by specifying the need for realistic and unique résumés, ensuring each résumé reflects a distinct candidate profile.

Phase 3 – Prompt Refinement and final Dataset Selection: In this phase, the optimal prompting approach was used to create a synthetic dataset, **df_synth_data**, based on 100 job descriptions (Appendix C.4). Testing this dataset on the baseline model achieved an F1-Score of 0.96, demonstrating the model's ability to distinguish between good and bad fit résumés. However, the high performance left limited room for further optimization. To address this, additional datasets were generated by refining prompts to better reflect real-world complexities and enable performance differentiation across model components (Test results in Table 4).

Dataset	***No. of job descriptions / résumés***	***Rows***	***Prompt for different fit résumés***	***Mean SC Diff***	***F1-Score***
df_synth_data	*100 / 6*	*600*	*„almost perfect "/ „slightly bad"*	*0.151*	*0.96*
df_synth_data_2_20	*20 / 6*	*120*	*„very good" / „moderate"*	*0.0179*	*0.6833*
df_synth_data_2_100	*100 / 6*	*600*	*„very good" / „moderate"*	*0.0169*	*0.62*
df_synth_data_3_20	*20 / 6*	*120*	*„almost perfect "/ „slightly bad to moderate"*	*0.0998*	*0.8833*
df_synth_data_3_100	*100 / 6*	*600*	*„almost perfect "/ „slightly bad to moderate"*	*0.1235*	*0.9067*
df_synth_data_4_20	*20 / 5*	*100*	*„perfect" / „good" / „moderate" / „bad" / „no" (Ranked 1-5)*	*N/A*	*0.51*

Table 4: Performance of different synthetic datasets on the baseline model: *The table summarizes synthetic datasets generated through refined prompts, detailing their composition, labelling strategies, mean similarity score differences (SC Diff), and F1-Scores. It highlights how prompt variations and dataset complexity affect baseline model performance.*

The final dataset, *df_synth_data_2_100*, was selected for its ability to better highlight areas for performance improvement, achieving a baseline F1-Score of 0.62. Figure 35 in Appendix C.5 consists of plots showing the distribution of similarity scores between good and bad fit résumés across the initial and final datasets. As shown, the baseline model performed notably worse at distinguishing between good and bad fit résumés on *df_synth_data_2_100*, indicating greater complexity and leaving room for improvement by other models. Consequently, this dataset offers a robust foundation for evaluating enhancements across the pipeline and testing the effectiveness of different model components.

It is important to note that the limitations associated with synthetic data, as previously discussed, also apply to the final dataset. Synthetic data may fail to replicate the nuanced patterns and expert-level granularity of real-world data. Additionally, biases inherent in the generative model could propagate into the dataset, potentially influencing downstream applications. These limitations highlight the need for careful interpretation of evaluation results and iterative improvements in the synthetic data generation process

7. Proof of Concept for Evaluating and Optimizing the Model Pipeline Using Synthetic Data

This chapter presents a comprehensive proof of concept for evaluating and optimizing the Skillmatch.ai model pipeline using synthetic data. It outlines the iterative benchmarking process designed to refine key pipeline components, focusing on skills representation, similarity thresholds, embedding models, and encoder architectures. Employing rigorous metrics such as F1-score, mean similarity score difference, and processing time, the evaluation highlights trade-offs between precision and operational efficiency. Insights gained inform the design of an advanced model pipeline, enabling precise matching while maintaining operational efficiency. This approach establishes a robust framework for model refinement, ensuring adaptability to real-world recruitment challenges and evolving business needs.

7.1 Evaluation Methodology and Objectives

The evaluation of the baseline model pipeline follows an iterative approach guided by the CRISP-DM framework, emphasizing continuous refinement to align technical outputs with business objectives (Wirth and Hipp 2000). This process focuses on optimizing key pipeline components while informing decisions on whether to deploy, refine, or reframe the model design (Hossain 2024, Raschka 2018).

The evaluation has three primary objectives. The first is to ensure precision in hiring decisions by accurately matching candidates to job requirements, minimizing errors such as false positives and negatives, enhancing workforce productivity, cost efficiency, and organizational reputation. The second objective is to optimize the model's performance for business operations by reducing processing time and ensuring scalability, making it suitable for seamless adoption in real-world recruitment processes. The third objective is to establish a framework for iterative refinement, enabling the model pipeline to tailor its performance to the specific needs of business operations. Once deployed in a company setup, the pipeline can incorporate real-world

data with feedback labels, overcoming data challenges and aligning outputs with business requirements. This approach ensures that businesses can evaluate and adopt emerging model components, remaining adaptive and competitive in the ever-evolving technological landscape.

7.2 Evaluation Process

This section provides an overview of the conducted evaluation process, beginning with an outline of the key components of the baseline model under assessment alongside the alternative solutions considered. It also discusses the metrics used to evaluate these components against their alternatives, focusing on achieving optimal performance and suitability for business operations. The final part explains how the setup is designed to ensure consistent and reliable results. All this provides a transparent overview and facilitates adjustments in future evaluation processes.

7.2.1 Overview of Benchmarking Components

As discussed in Chapter 5, the components of the baseline model were selected based on a theoretical perspective during the model design phase. While pdf parsing and skill extraction have demonstrated strong performance, this section shifts focus to the key concepts around the quantification of skills through embeddings, which enable the similarity analysis. Across four iterations, the following components will be evaluated.

1. Iteration - Format of Skill Representation for Embeddings: With the baseline approach, Bag-of-Skills, all candidate skills are aggregated into a single vector representation. This method was expected to perform well by capturing interrelations among skills (Chapter 5.2.6). In contrast, the Skill-by-Skill format processes each skill individually, potentially facilitating enhanced skill alignment (Dridri 2022). This iteration assesses the trade-offs between these two approaches, focusing on their impact on precision in skill matching.

2. Iteration - Hyperparameter Selection: Building upon the skill-by-skill approach, the next step evaluates the threshold hyperparameter in the similarity calculation process. This threshold

defines the minimum cosine similarity score required for two skills to be considered a match. In the process of similarity score calculation every skill listed in the résumé is compared with each skill of the job description. Exact matches contribute 1 to the score, while non-exact matches contribute their cosine similarity value only if their highest similarity score meets or exceeds the threshold. Dridri (2022) used this strict equality check with a cosine similarity threshold of 0.4. To optimize the threshold in a skills-by-skill approach, values of 0.3, 0.4, 0.5, and 0.6 are compared towards their performance improvements.

3. Iteration – Embedding Models: This iteration benchmarks the baseline embedding model (all-mpnet-base-v2) against three additional models: one other build further upon the SBERT framework (Reimers and Gurevych 2019) and two state-of-the-art models ranked on the Massive Text Embedding Benchmark (MTEB) leaderboard (Muennighoff, et al. 2023).

1) all-mpnet-base-v2 (baseline): Built on Microsoft's MPNET pre-training strategy (Song, et al. 2020), this model balances computational efficiency and semantic accuracy, making it a widely adopted baseline embedding model (Muennighoff, et al. 2023). **2) all-MiniLM-L6-v2**: Based on Microsoft's MiniLM (Wang, et al. 2020) this compact SBERT model is optimized for efficiency, maintaining high performance while reducing computational demands (Muennighoff, et al. 2023). 3) **NV-Embed-v2**: NVIDIA developed the top-performing model on the MTEB leaderboard excelling in semantic accuracy but requiring extensive GPU resources (Lee, et al. 2024; Moreira, et al. 2024). **4) SFR-Embedding**: Fine-tuned on E5-mistral-7b-instruct, this Salesforce model balances precision and flexibility, ranking 5th on the MTEB leaderboard (Meng, et al. 2024)

Embedding Model	***Model Size (Million Parameters)***	***Embedding Dimension***	***Memory Usage (GB, fp32)***
all-mpnet-base-v2	*110*	*768*	*0.41*
all-MiniLM-L6-v2	*33*	*384*	*0.12*
NV-Embed-v2	*7851*	*4096*	*29.25*
SFR-Embedding	*7111*	*4096*	*26.49*

Table 5: Model Characteristics ("MTEB Leaderboard - a Hugging Face Space by Mteb," November 28, 2024): *Illustrates that larger model, such as NV-Embed-v2 and SFR-Embedding, require significantly more memory and computational resources compared to smaller, more resource effective models like all-MiniLM-L6-v2 (Muennighoff, et al. 2023).*

These significant differences in model characteristics, as shown in Table 5, are leveraged to benchmark the baseline model. This process provides valuable insights into whether state-of-the-art models and larger model sizes impact the overall performance of the Skillmatch.ai pipeline.

4. Iteration - Encoder Architectures: The baseline model employs a bi-encoder architecture, independently encoding inputs (résumé and job description skills) into fixed-size embeddings for similarity comparison. In contrast, cross-encoders jointly process input pairs, expected to catch richer interactions between inputs (see Chapter 5.2.6 for details). While cross-encoders have shown superior performance on certain tasks (Reimers and Gurevych, 2019) their computational demands are significantly higher. This iteration evaluates the two architectures, comparing their computational efficiency and performance outcomes under testing conditions.

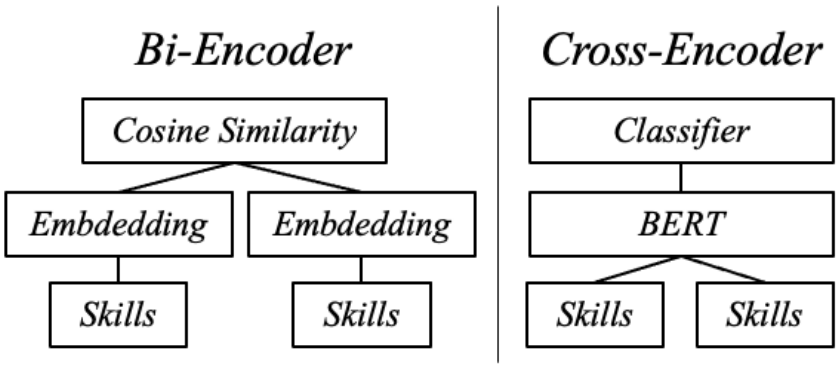


Figure 9: Encoder Architectures: *Comparing Bi-Encoder Architecture with Cross-Encoder Architecture*

7.2.2 Evaluation Metrics

To assess the performance of the Skillmatch.ai pipeline, we utilize binary classification metrics (Precision, Recall and F1-Score), processing time, and the mean of similarity score difference between labels. These metrics balance predictive precision and operational efficiency, addressing the practical demands of recruitment systems. See Appendix D.1 for further details.

Binary classification metrics provide a framework for evaluating model predictions in recruitment, where errors such as false negatives (overlooking qualified candidates) or false

positives (recommending unsuitable ones) can have significant consequences. Research highlights their importance in classification tasks, emphasizing their practical relevance in the systematic assessments of accuracy (Powers 2020; Goutte and Gaussier 2005). The **F1-Score** is our primary evaluation metric. Calculated as the harmonic mean of precision and recall it balances correctly identifying suitable candidates with minimizing overlooked talent. This makes it especially suitable for recruitment, where precision and recall are critical to reliable hiring decisions.

Processing time is a vital metric for evaluating efficiency. Faster models reduce recruitment timelines, streamline workflows, and enhance candidate experience. By minimizing processing time, organizations can improve decision-making speed while maintaining predictive accuracy.

The Mean Similarity Score Difference evaluates the model's ability to distinguish between Good Fit (Label 1) and Bad Fit (Label 0) candidates by calculating the average difference in similarity scores. Higher Mean SC Difference values indicate clearer distinctions, improving the reliability of candidate ranking and the overall quality of recommendations.

7.2.3 Evaluation Setup

To ensure consistent benchmarking, all evaluations were conducted in a Python notebook under identical conditions, using a stable Wi-Fi connection and the NVIDIA A100 GPU within a Google Colab environment. The entire evaluation process is documented in the GitHub repository (Appendix D.2). This controlled setup minimized external variability, ensuring that differences in processing time and performance reflected model configurations alone. The synthetic dataset df_synth_data_2_100 was used across all evaluations for comparability. With the lowest baseline model performance during initial testing (F1-Score of 0.62), this dataset was chosen to highlight improvements from different model configurations more reliably (see Section 6.4 for detailed discussion). To address the computational demands of NV-Embed-v2 and SFR-Embedding, which require significantly higher GPU memory (Appendix D.2.3) the

test dataset was reduced to 120 samples (20 job descriptions with 6 résumés each). This adjustment prevented kernel crashes, ensured faster evaluations, and maintained sufficient complexity to analyse the impact of embeddings on matching accuracy and processing efficiency.

7.3 Results

Coming to the results, this section summarizes the benchmarking outcomes for the Skillmatch.ai pipeline. With each iteration, the best-performing components were integrated to refine the pipeline. Detailed results and visualizations can be found in Appendix D.2.

7.3.1 Format of Skill Representation for Embeddings

In the first iteration, we evaluate the impact of skill representation formats - Bag-of-Skills (BOS) versus Skill-by-Skill (SBS): The results demonstrate that SBS significantly outperforms BOS. SBS achieves an F1-Score of 0.7471, effectively identifying good-fit résumés and minimizing false positives. In comparison, BOS achieves an F1-Score of 0.62, highlighting its limitations in distinguishing between good-fit and poor-fit candidates.

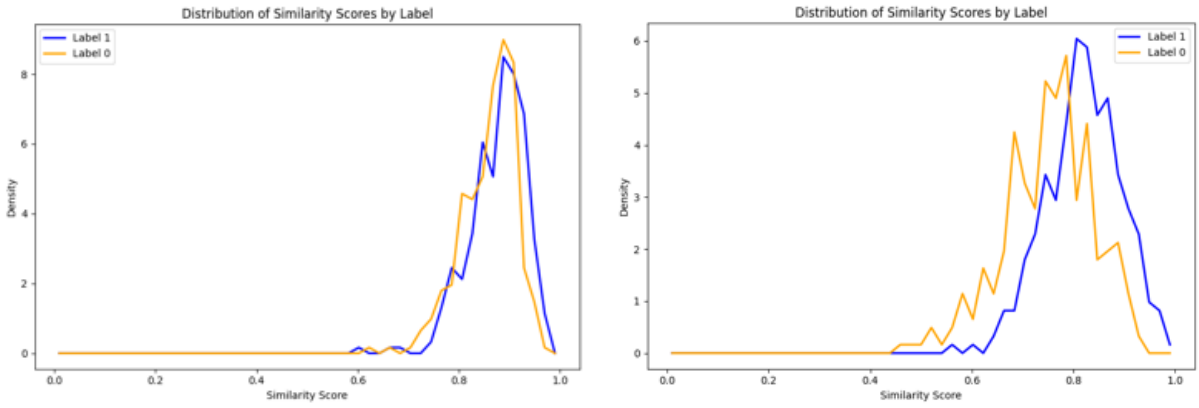


Figure 10: Distribution of Similarity Scores for Skill Representation Formats: *This figure compares the similarity score distributions for Bag-of-Skills (left) and Skill-by-Skill (right). It shows that the Skill-by-Skill approach produces a clearer separation of scores between good-fit (Label 1) and bad-fit (Label 0) résumés.*

Additionally, the Mean Similarity Score Difference for SBS is 0.0675, compared to just 0.0169 for BOS, indicating a much clearer separation between good-fit (Label 1) and bad-fit (Label 0) résumés. Figure 10 visually supports this finding, showing that SBS produces a better separation of similarity scores, while BOS exhibits significant overlap. While SBS requires a longer processing time of 550.95 seconds, the BOS approach completes processing in 138.66 seconds, making it a faster option. However, this speed comes at the cost of reduced performance, which limits its ability to differentiate between candidates effectively. The trade-off between accuracy and computational cost reinforces SBS as the preferable format for subsequent iterations, where further optimization will focus on hyperparameter selection.

7.3.2 Hyperparameter for Skill-by-Skill Similarity Calculation

In this iteration, the focus is on optimizing the threshold hyperparameter used in the SBS similarity calculation. Threshold values of 0.3, 0.4, 0.5, and 0.6 were evaluated to determine their impact on model performance and computational efficiency. The results show that a threshold of 0.4 achieved the best overall performance, yielding an F1-Score of 0.7471. This setting balanced precision and recall while maintaining a reasonable processing time of 550.95 seconds and achieving a Mean Similarity Score (SC) Difference of 0.0675, indicating strong class separation between good-fit (Label 1) and bad-fit (Label 0) résumés. Higher thresholds, such as 0.6, resulted in a slightly larger Mean SC Difference (0.0861), suggesting stricter matching criteria with better separation (Appendix D.2.2). However, this came at the cost of a lower F1-Score, as stricter thresholds excluded some good-fit résumés, reducing overall predictive performance. These findings validate the threshold of 0.4 as the optimal choice, striking a balance between class separation, computational efficiency, and predictive performance.

7.3.3 Embedding Model

The results for the third iteration, shown in Table 6, highlight the performance and efficiency trade-offs among the evaluated embedding models. Detailed results in Appendix D.2.3.

Embedding Model	*Model Size (Million Parameters)*	*Embedding Dimension*	*Memory Usage (GB, fp32)*	*Processing Time*	*Mean SC Difference*	*F1-Score*
all-mpnet-base-v2	*110*	*768*	*0.41*	*140.80s*	*0.0631*	*0.6723*
all-MiniLM-L6-v2	*33*	*384*	*0.12*	*106.30s*	*0.0623*	*0.6838*
NV-Embed-v2	*7851*	*4096*	*29.25*	*5030.74s*	*0.0354*	*0.6891*
SFR-Embedding	*7111*	*4096*	*26.49*	*5306.20s*	*0.0157*	*0.6667*

Table 6: Comparison of Embedding Model Characteristics and Performance Results: *This table highlights the trade-offs between performance and operational efficiency for the evaluated embedding models, showcasing differences in F1-Score, processing time, and resource usage.*

NV-Embed-v2 achieved the highest F1-Score (0.6891), but its impractically high processing time of 5030.74 seconds and significant memory usage make it unsuitable for scalable applications. Similarly, SFR-Embedding took even longer to process (5406.20 seconds) and delivered the lowest F1-Score (0.6667) with a poor Mean Similarity Score Difference of 0.0157, limiting its ability to distinguish between good and bad fits. By contrast, all-MiniLM-L6-v2 emerged as the optimal choice, providing the best balance of performance and efficiency. It achieved an F1-Score of 0.6838, the shortest processing time of 106.30 seconds, and a Mean SC Difference of 0.0623, which significantly enhances class separation. The baseline model, all-mpnet-base-v2, showed comparable performance with an F1-Score of 0.6723 and a Mean SC Difference of 0.0631, but required a longer processing time of 140.80 seconds. These results confirm all-MiniLM-L6-v2 as the most efficient embedding model, offering time-effective scalability and precise recruitment operations, making it ideal for real-world deployment.

7.3.4 Transformer Architectures

In the last iteration, we evaluate the performance and efficiency of the used Bi-Encoder against the Cross-Encoder Architecture. The results show that the Cross-Encoder achieves an F1-Score of 0.74, comparable to the Bi-Encoder, which performs slightly better with an F1-Score of 0.7407. However, the Cross-Encoder's processing time of 4342.99 seconds (over 5 times longer

than the Bi-Encoder's 749.35 seconds) reveals a significant trade-off. This substantial processing time severely limits the feasibility of the Cross-Encoder for scalable recruitment systems where efficiency is critical. While the Cross-Encoder offers competitive performance, its excessive computational cost makes it unsuitable for real-world deployment in the Skillmatch.ai pipeline. These findings validate the Bi-Encoder architecture as the preferred choice, delivering reliable results while ensuring operational efficiency to meet the demands of modern recruitment systems. For detailed result of this iteration, see Appendix D.2.4.

7.4 Conclusion: Advanced Model Pipeline

The evaluation process provided a clear learning roadmap for optimizing the skill-first matching model pipeline. Starting with the Bag-of-Skills approach, we advanced to the Skill-by-Skill representation, where a threshold of 0.4 delivered the best performance. In the embedding model evaluation, all-MiniLM-L6-v2 emerged as the best-performing option, balancing prediction performance and processing time efficiency. The Bi-Encoder architecture, as the foundation of the pipeline, was further validated for its ability to deliver results six times faster than the alternative Cross-Encoder approach without compromising performance.

Final testing on a larger dataset revealed an unexpected processing time anomaly for all-MiniLM-L6-v2. Upon re-evaluation alongside all-mpnet-base-v2, both models exhibited similar processing times, but all-MiniLM-L6-v2 demonstrated a slight edge in distinguishing between good-fit and bad-fit résumés. Its performance on the smaller datasets and lower computational demands confirmed its suitability as the embedding model for the pipeline.

These findings collectively resulted in an advanced model pipeline that delivers more accurate candidate-job matches while minimizing false positives and negatives. Additionally, operational efficiency was improved by optimizing processing time and ensuring scalability. With this process, a framework for iterative refinement was established, allowing the model to adapt to evolving business needs and real-world recruitment challenges.

7.5 Limitations

While the evaluation process provided valuable insights, limitations remain. The use of synthetic datasets ensured consistency but may not fully replicate the complexity of real-world résumés and job descriptions, limiting the precision objective of the evaluation. Transitioning to real-world data will introduce challenges such as inconsistent skill naming and domain-specific nuances, requiring further validation to ensure accurate skill matching.

7.6 Future Outlook

The next step focuses on transitioning from synthetic datasets to real-world company data, which will provide more tailored and context-specific solutions. However, this shift introduces new challenges, such as dealing with inconsistencies, noise, and variability in real résumés. Developing a robust data collection strategy, supported by partnerships with organizations, will be essential to ensure diverse and representative datasets. Additionally, addressing ethical considerations becomes even more critical when working with real-world data. Ensuring fairness, mitigating biases in skill representation, and maintaining transparency will remain key priorities. Lastly, fine-tuning the embedding models with company-specific skills data offers promising opportunities but will require careful validation to ensure precision and adaptability in real-world recruitment scenarios.

8. Skills-Based Similarity Model for Career Development

Evolving job market dynamics necessitate a paradigm shift towards skills-based career development. This analysis is a proof of concept for developing a ranking system that suggests potential role transitions by mapping individuals' skills to roles with overlapping competencies based on their résumés. The resulting tool enables individuals to make informed career decisions and assists organizations in strategic talent management. By identifying key challenges and offering a solid foundation for future advancements, this study supports ongoing efforts to refine skills-first practices and promote their integration into talent management strategies.

8.1 The Importance of Identifying Skill-Aligned Career Pathways

The World Economic Forum's *Future of Jobs Report 2023* predicts 69 million new jobs and the loss of 83 million jobs over the next five years (Di Battista, et al.2023). In this rapidly evolving landscape, individuals must identify clear career development pathways to support lifelong learning and adaptability (ibid.), essential to remain employable and achieve career growth. On the other side, organizations should invest in internal talent development to boosts engagement, retention (Adewummi 2024) and workforce productivity (Mustafaa and Lleshi 2024) and to remain competitive in a changing labour market (Subramaniam, Talib and Khan 2018). However, individuals face challenges in identifying skill-aligned opportunities due to inaccurate perceptions of their skills (Kiss, et al. 2023) and inconsistent job descriptions (Stybel 2010). Organizations need significant effort and resources to detect which employees to prioritize (Cappelli and Keller 2014) and to assess skills mismatches (Saad, et al. 2023), further complicated by the lack of standardized competency systems (Moses 2016). Given these challenges, this section explores the following inquiry: *"How can we leverage skill overlaps to develop a tool that facilitate career transitions into closely related roles, supporting a skills-first approach?"*. In addressing this question, the analysis provides a proof of concept,

contributing to the development of the Skillmatch.AI ecosystem. This model ranks the closest matching roles by comparing the skills listed in a résumé with the skill requirements of various positions. Further details on the methodology are given in paragraph 8.4. The in-depth exploration of the steps, challenges, and limitations involved in developing such a tool offers insights and practical guidance for future efforts. By contributing with this proof of concept and a detailed examination of the development process, this study seeks to foster the adoption of skills-first practices and supports the broader transition towards skills-based career development.

8.2 Skill-Based Role Recommendations Objective

Role recommendations are based on similarity calculations between the individual's skills and the role requirements. This approach, while similar to the model proposed in Chapter 5, shifts the perspective from hiring to talent development. Instead of matching candidates to a single job, it identifies multiple potential roles for an individual. The goal is to enable data-driven decisions for career transitions by ranking potential roles based on their alignment with a person's skills. Based on a **skills-first approach**, the similarity model aims to improve talent development and career planning for both individuals and organizations. Job seekers can identify roles that match their skill set, targeting opportunities that maximize success (European Labour Authority 2023). Companies can optimize internal mobility by transitioning employees in obsolete roles to in-demand positions by identifying skill overlaps. At the same time, HR departments can design career paths and training programs that align with the evolving demands of the workforce (Kaminski 2024).

8.3 Data Used

Two databases from O*NET (O*NET Resource Center 2024) were used to define roles.

"**Technology Skills**" highlights the most frequently mentioned skills in job postings, categorized into Hot Technologies and In-Demand Skills. Only skills classified by O*NET as

both *in-demand* (critical for specific roles) and *hot technologies* (highly relevant to the current job market) were considered. This approach ensures that the similarity model emphasizes skills valuable in today's labour market, helping candidates and organizations navigate high-demand and rapidly evolving contexts.

The second table from O*NET, "**Occupation Data**", provides general roles descriptions. Incorporating this data allows the model to account for broader concepts, such as "marketing", "accounting" or "web development". This addition enhances the model's robustness by enabling similarity calculations to go beyond technical terms, improving the alignment between résumé skills and role requirements. More information about the exploration and cleaning of the datasets is provided in Appendix E.1.

O*NET offers multiple dimensions for defining roles, such as activities, abilities, and tasks. Initially, some of these features were considered, however, it was observed that these attributes are often described inconsistently by candidates, making them difficult to standardize for comparison. In contrast, hard skills, despite slight variations in terminology, tend to remain uniform across résumés and roles. This consistency allows for a more reliable and objective comparison, ensuring that similarity calculations accurately reflect the alignment between a candidate's skills and job requirements.

8.4 Methodology for Skill-Based Similarity Calculation

To generate role recommendations, the similarity model takes two inputs: a résumé vector and vectors for all 453 roles sourced from O*NET. The analysis begins with extracting skills from résumés using **spaCy's Entity Ruler** (Honnibal and Montani 2017) and a custom pattern file to create the required vector (section 5.2.4). Each role's vector is then defined by two components: keywords extracted from the role description, applying same approach as above, and hard skills sourced from O*NET data. Table 7 displays examples of role vectors. These résumé and role vectors are passed through the model and converted into embeddings using the **MPNet model from HuggingFace (**Devlin, et. al. 2019**),** as detailed in section 5.2.5. Finally,

Cosine similarities (Formula 3) are calculated, as explained in section 5.2.6, to measure the alignment between the résumé and each role (Sohangir and Dingding 2017), resulting in a ranked list of role recommendations. These steps are visualized in Appendix E.2 and code of the model is provided in Appendix E.4.

Title	***Skills***
Mechanical Engineers	*['Autodesk AutoCAD', 'Autodesk Revit', 'Microsoft PowerPoint', 'Microsoft Excel', 'Python', 'Dassault Systems SolidWorks', 'Microsoft Outlook', 'engineering']*
Market Research Analysts	*['Google Analytics', 'Microsoft PowerPoint', 'search engine', 'Microsoft Excel', 'marketing', 'advertising', 'operations research', 'Microsoft Outlook', 'Salesforce software']*
Computer and Information Systems Managers	*['SQL', 'Microsoft Azure software', 'computer programming', 'Amazon Web Services AWS software', 'data processing', 'Microsoft Excel', 'SAP software']*

Table 7: Example of list of keywords per each role collected from O*NET: *This table shows role vectors represented by both hard skills and occupation keywords. This dual representation offers richer information and allows for a more accurate and comprehensive depiction of each role.*

While the extraction, embedding, and similarity analysis processes remain the same, the output of this model differs from the version presented in Chapter 5. The output is a table (Table 8) where each row represents a role along with its ranking. Additionally, the model includes features such as missing skills, salary differences, and the role's growth potential, offering greater value to the end user. Their integration is detailed in Appendix E.3.

Rank	***Title***	***Missing Skills***	***Annual Wage Difference compared to actual role***	***Role Information***
1	*Market Research Analysts*	*Salesforce Software*	*+ 8190*	*Rapid Growth*
2	*Supervisors of Sales Workers*	*Salesforce Software*	*+ 24020*	*Not In-Demand*
3	*Management Analysts*	*Tableau, Microsoft Power BI*	*+ 40530*	*Numerous Job Openings*

Table 8: Final Output: *This table presents the closest roles identified by the similarity model, along with key insights to inform career transitions. For each role, we see missing skills required for the transition, potential salary variation (increase or decrease), and the role's market demand (e.g., growth prospects or number of job openings). This comprehensive overview helps users make informed decisions by considering not only role similarity but also other critical factors influencing career choices.*

The custom pattern file for word extraction, detailed in section 5.2.4, was increased by 119 additional technical skills from O*NET data and new patterns to capture them from résumés in various formats. This updated version can be found in "*Resources/data/skills_branch.jsonl"*

directory of our codebase. However, this enhancement does not fully resolve **a major limitation**: résumés often lack complete information and describe skills in inconsistent ways. This variability, combined with the constraints of a predefined dictionary, limits the number of identifiable keywords. Achieving consistency between the number of keywords extracted from résumés and those defining the roles is essential for ensuring more accurate and balanced similarity calculations.

The following paragraphs describe a three-step process used to develop the final version of the model. In the first step, similarity scores were calculated for vectors created by hard skills and occupation keywords separately, providing a clear comparison of results across datasets. Secondly, hard skills and occupation keywords were combined to create role vectors, offering a more comprehensive representation. Finally, the third step explored integrating the number of matching words into the semantic similarity scores to refine the results.

8.4.1 Separated similarity scores

The initial similarity calculations were conducted separately for the two datasets, **Technology Skills** and **Occupation Data**. By isolating specific components, it is possible to evaluate their individual contributions to the similarity calculation, laying the basis for more complex refinements. While this approach provided an initial baseline, significant limitations emerged due to the characteristics of the two datasets.

As shown in Figure 11 and Figure 12, most of the roles are defined by only a few keywords/hard skills. For instance, 149 roles had fewer than three words extracted from role descriptions. This sparsity limits the ability of the datasets alone to provide a robust representation of roles, as there is insufficient information to differentiate among them. Consequently, cosine similarity scores may be inflated: with fewer skills being compared, it becomes easier for the résumé's embedding to appear similar to the role's embedding, even if they are not the best match for the individual's profile.

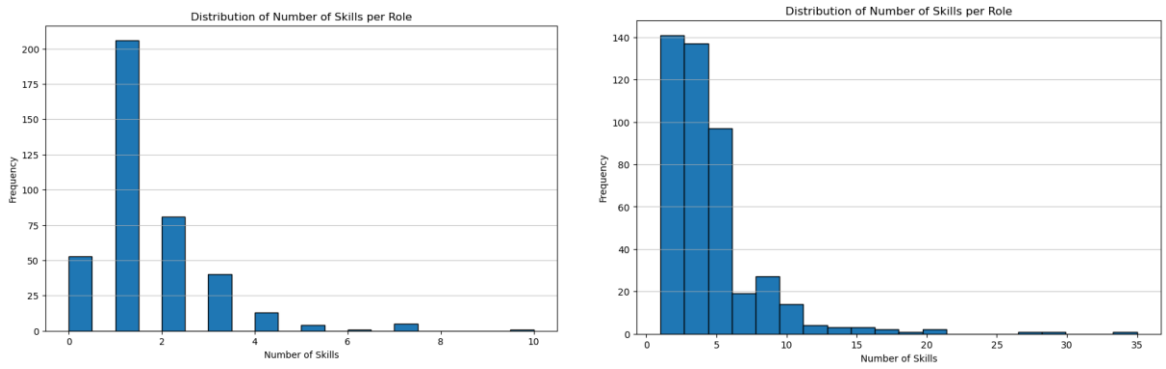


Figure 11: (on the left) Distribution of number of occupation keywords per role: *the bar chart illustrates that most roles are defined by just one keyword, with only a few roles having more than five keywords. This highlights the sparsity of role representation based on Occupation Data, underscoring the challenge of capturing comprehensive skill profiles.*

Figure 12: (on the right) Distribution of the number of hard skills per role: *while some roles are characterized by a significantly higher number of skills, with a few having more than fifteen, the bar chart shows that most roles are defined by just two to five hard skills. This highlights the overall tendency for roles to be represented by a limited set of skills. This variability underscores the challenge of creating comprehensive and consistent skill profiles across different roles.*

While each dataset offers valuable insights, neither is sufficient on its own to capture the complexity of role requirements comprehensively. This analysis paved the way for more integrated approach that merge both sources of information for better alignment.

8.4.2 Merged information for a comprehensive similarity score

In this version, hard skills (in-demand and hot technologies) and keywords from role descriptions (from the Occupation Data file) are merged to provide a more comprehensive representation of each role. Adding context through role-specific keywords allowed the embeddings to better capture the nuanced application of hard skills across different professions – at this point, the model distinguishes how WordPress is applied in a marketing role versus its use in a Web Developer role.

The merged information led to higher similarity scores for résumés with fewer technical skills, as the contextual information provided a better representation of these roles. Conversely, for résumés with a higher number of hard skills, these already captured the core technical requirements, leaving less room for improvement. Indeed, hard skills contribute much more than keywords when defining the vectors per role (Figure 13).

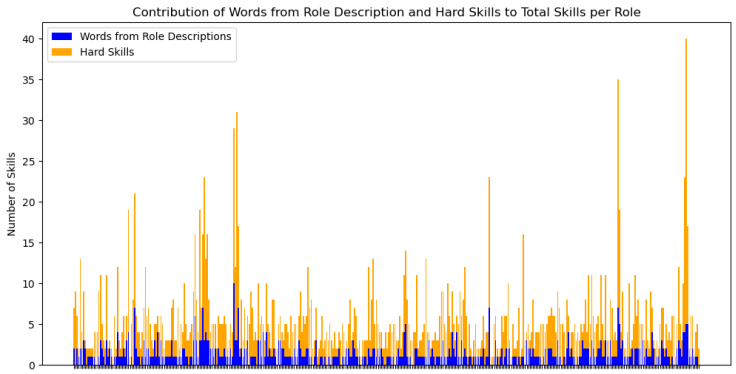


Figure 13: Contribution of hard skills and words from role descriptions: The bar chart shows that hard skills (orange bars) contribute significantly more than keywords from role descriptions (blue bars) when defining role vectors. This indicates that hard skills play a larger role in representing the skill profiles of each role.

While merging the datasets provided a more **holistic representation** of roles, this does not fully resolve the issue of sparsity in role descriptions. While adding more information from O*NET increases the word count for roles, the word count for résumés cannot be similarly expanded. Given the incomplete data in résumés, similarity scores can be misleading, regardless of the number of words representing the roles.

Additionally, in some cases, the similarity model assigns higher scores to roles with fewer matching skills. This is due to how the embeddings represent the entire set of terms for each role. Research from Netflix demonstrates that cosine similarity of learned embeddings can yield arbitrary results. They explain that the issue lies not with cosine similarity itself, but with the degree of freedom in the learned embeddings, which can produce inconsistent cosine similarity scores (Steck, Eksanadham and Kallus 2024).

8.4.3 Incorporating number of matching words into the Similarity Score

To address this limitation, this section proposes three approaches that combines semantic similarity with numbers of matching words to enhance the precision of role alignment.

The Scoring Approach uses a strict equality check combined with cosine similarity threshold of 0.4, comparing skills on a skill-by-skill basis. The logic is presented below. This method is adapted from the approach proposed by Amira Dridri (Dridri 2022).

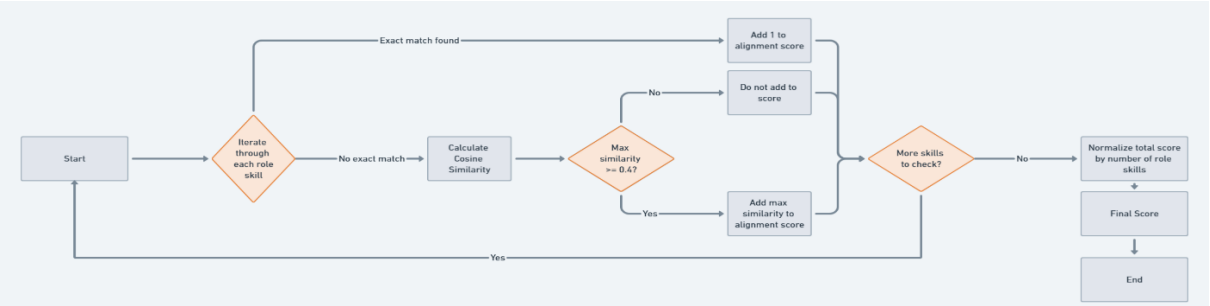


Figure 14: The Scoring Approach: *This approach is based on calculating a score based on exact matches and cosine similarity.*

When applying this method, the scoring system disproportionately favoured roles with exact word matches, even when cosine similarity is low. This issue is particularly problematic when there are few skill requirements for a role, allowing for an easy perfect match. As this method does not adequately balance the contributions of exact matches and semantic similarity it was ultimately excluded.

Proportional Approach calculates the proportion of matching words relative to the total list length and then it is multiplied by similarity score.

$$Proportional\ Score = \text{Cosine similarity} \times \left(\frac{\text{Number of matching words}}{\text{Total Number of Skills in a Role}}\right) \quad (6)$$

After testing and evaluating the Proportional Approach, it was ruled out to three significant issues. First, normalizing the number of words matching by the total number of skills for a role penalized roles with longer skill lists. For instance, a role with three skills and two matches (66%) scores higher than one with fifteen skills and five matches (33%), despite the latter having stronger alignment. Second, multiplying the normalized word match proportion by the cosine similarity often resulted in disproportionately low scores for roles with few matches but high cosine similarity. Finally, when no word matches are found, the final score is reduced to zero, eliminating the contribution of semantic similarity.

Weighted Approach calculates the proportion of matching words relative to the total number of words in the role's vector and multiplies it by a designated weight. The product is then summed up with the semantic similarity score, weighted by the remaining proportion.

$$\textit{Weighted Score} = \text{Cosine similarity} \times \text{w} + \left(\frac{\text{\# of matching words}}{\text{Total \# of Skills in a Role}}\right) \times (1 - w) \qquad (7)$$

To determine the weights, various configurations were tested, starting from 0.5 (equal importance) and increasing the semantic similarity weight by 0.1 up to 0.9. The goal was to prioritize semantic similarity while using word matching to refine rankings. In most cases, adjusting the weights produced minimal changes to the rankings. However, some roles that were initially ranked low based on semantic similarity jumped to top positions, causing ranking distortions. The lack of a benchmark made it difficult to identify the optimal balance; however, true rankings by experts would allow for a systematic grid search to fine-tune the weights (Liashchynskyi and Liashchynskyi 2019). Alternatively, synthetic data could be used, but this approach was not pursued due to complexity and time constraints.

In summary, without clear evidence demonstrating superior performance with the Weighted Approach, relying solely on cosine similarity is a better alternative. It reduces complexity and maintains focus on the contextual alignment that is the most reliable indicator of role relevance.

8.6 Limitations and Further Development

As highlighted during the analysis, this process revealed two main limitations: 1) résumés have inconsistent descriptions and therefore extracting enough skill can be hard; and 2) the embedding process can influence cosine similarity scores, sometimes resulting in arbitrary outcomes.

To improve skill extraction, the custom pattern file could be expanded to capture even more skills. Ideally, skills should be presented uniformly in both résumés and role descriptions, as capturing every possible variation in how a skill is mentioned can be highly complex. This remains quite an issue since the way skills are presented depends on individual choices.

Involving career counsellors to identify the best role recommendations for each résumé would create a ground-truth dataset for benchmarking the model's performance. This would allow different embeddings and similarity metrics to be tested, possibly mitigating the second limitation by identifying the most effective approach. Metrics such as Mean Squared Error (MSE) can help minimize errors in similarity scores, while Precision@K**,** Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP) can evaluate how well the model ranks relevant roles (Luchen and Clearke 2014). These metrics are further explained in Appendix F.3. Though generating synthetic data could theoretically simulate this scenario, time constraints and complexity made it impractical within the scope of this study.

8.7 Key Insight

This analysis aims to foster the adoption of a skills-first approach, not only in recruitment but also in post-hiring processes, such as internal role transitions and career development. To achieve this, the study addressed the question: *"How can we leverage skill overlaps to develop a tool that, supporting a skills-first approach, facilitates career transitions into closely related roles?"*. The primary goal was to analyse the steps involved in building such a tool, where the key takeaway lies in the detailed examination of these challenges. Given the absence of robust evaluation methods, the focus remains on understanding and addressing the current model's limitations to pave the way for more sophisticated skills-first tools. This analysis offers practical insights for refining future models and highlights areas needing improvement, contributing to the development of the **Skillmatch.ai** ecosystem.

9. Enhancing Role Recommendations through Comprehensive Occupational Similarity Analysis

*This section proposes a role-to-role matching approach that moves beyond relying solely on résumés by incorporating broader occupational dimensions, including tasks, knowledge, skills, abilities, and detailed work activities from O*NET. Correlation and R-squared analyses guide the feature selection, and the final model integrating KSAs, tasks, and DWAs shows improved alignment with O*NET-based benchmarks. The resulting network analysis reveals clusters of occupations sharing common attributes, underscoring the connected nature of different roles. However, limitations such as low explained variance and reliance on predefined criteria persist. Further refinements, such as expert validation, could enhance the model's robustness. Ultimately, this approach lays groundwork for more comprehensive, skill-oriented role recommendations.*

9.1 Motivation

The previous section proposed a tool for ranking the most similar roles for a given résumé. However, this approach faces several challenges. First, résumés often provide incomplete information and are influenced by how candidates choose to present their skillset. Second, role definitions tend to focus primarily on technical skills, neglecting important dimensions such as soft skills and tasks performed. As a result, these gaps can lead to limited skill representation in both résumés and roles, yielding suboptimal role recommendations.

To address these limitations, this section proposes an additional component that captures role similarity on a more general basis. This approach moves beyond simply matching technical skills found on a résumé. Instead, it evaluates the broader skillsets, knowledge, tasks, and attributes typically associated with a role, by creating a role-to-role matching model. The assumption is that individuals in a specific occupation are likely to possess core competencies and qualities relevant to that role, even if these are not explicitly mentioned in their résumés.

This section aims to answer the following question: *"How to overcome the limitations of résumé-based role ranking and enhance the accuracy and comprehensiveness of role recommendations?"*

To build this model, several steps where undertaken. First, **data was collected from O*NET's comprehensive dataset** defining roles across dimensions such as tasks, abilities, knowledge, activities and values. Next, **the most relevant dimensions were selected as features** for the ranking model, guided by previous research and evaluated through R-squared and correlation analysis. The model's performance was then **validated using the O*NET dataset on related occupations.** Finally, the role-ranking results were **visualized through network analysis** to illustrate role relationships and similarities. The final output is a role-to-role ranking model that supports talent development and career decisions by offering generalized recommendations. Integrated into the web application alongside the résumé-role ranking model, it provides comprehensive insights by considering both individuals' technical hard skills and broader role dimensions.

9.2 Data and Feature Analysis

Traditional frameworks suggest identifying role similarity based on three attributes that collectively define competence - skills, abilities, and knowledge (KSAs) (Hadzilacos, et al. 2019; Allen, et al. 2012). In contrast, more recent O*NET research prioritizes the similarity of actual work performed - tasks and detailed work activities - over more abstract worker-related characteristics (Dahlke, et. al 2022). Building on both perspectives, this analysis expands the scope to include KSAs, tasks, and detailed work activities when evaluating occupational similarity. The table below summarizes the O*NET data elements used.

Table Name	Foundation for Similarity Measure	Scale	Example
Skills	Numerical	Importance (1-5), Level (0-7)	Programming, Importance: 4.67
Knowledge	Numerical	Importance (1-5), Level (0-7)	Law & Government, Importance: 2.62
Abilities	Numerical	Importance (1-5), Level (0-7)	Oral comprehension, Importance: 1.3
Tasks	Text	Core / Supplemental	Collect data through direct observation of work activities
Detailed Work Activities	Text	Not Applicable	Negotiate contracts

Table 9: Overview of Collected O*NET Tables: *The table illustrates the O*NET data elements considered as feature for the role-ranking model - skills, knowledge, abilities, tasks and detailed work activities - along with their key attributes: table name, data format, scale and example values.*

To determine which features to include in the final occupation recommendation model, an evaluation process was conducted. For each feature, role similarities were calculated using similarity measures based on the data's format (e.g., numerical or text). These features were then evaluated individually and in combination to assess their effectiveness as predictors.

Evaluation Through Correlation and R squared Analysis: Each feature was assessed against predefined criterion to evaluate its relevance for role similarity. The assumption is that occupations sharing similarities in KSAs, tasks, or activities (features) are also likely to align in other domains, such as titles, salary, work styles, work values and work interests (criteria). Correlation analysis measured the strength of the relationship between features and criteria (Schober 2018), while R-squared analysis quantified how much variance in the criteria could be explained by these features (Chicco, Warrens, and Jurman 2021). This approach was chosen as it aligns with methodologies established in prior research, where O*NET applied similar technique but with different criteria (Dahlke, et al. 2022).

Benchmarking Against O*NET Rankings: The final model was evaluated by benchmarking the ranking against O*NET dataset on occupational similarity. Ranking measures such as

Precision@K, Mean Reciprocal Rank and Mean Average Precision were deployed (Konstantinidis et. al, 2022).

9.3 Role Similarity Calculation

Knowledge, skills, and abilities (KSAs) were combined into a single composite similarity measure, as KSAs capture foundational attributes that distinct from more task- or activity-specific metrics (Mototwidlo, Borman and Schmit 1997). Each KSA include two numerical values in the dataset: importance and level. While previous research has used both (Hadzilacos, et al., 2019) for similarity calculations, precision issues were identified with level values. Therefore, only importance values were selected for calculating similarity scores. See Appendix F.1 for scale definitions and precision details.

Similarities were calculated using the squared euclidean distance (Appendix F.1) of importance ratings across KSAs for each role pair because it captures both the magnitude and variance of differences in importance ratings, rather than focusing solely on their relative order (Dokmanic, et. al 2015). Z-scores were computed separately for the euclidean distances in KSAs and then summed to form a final composite similarity score, providing a balanced representation of occupational similarity (Allen, et al. 2012).

$$KSA\ Distance\ Compisite = Z\left(D_{knowledges}\right) + Z(D_{Skills}) + Z\left(D_{Abilities}\right) \quad (8)$$

Tasks and Detailed Work Activities (DWAs) were converted to embeddings using the MPNet model from Hugging Face. This method was selected for its strong performance in the baseline model (section 7.2.1). Role embeddings were created by averaging the embedding vectors across tasks and activities and cosine similarity (defined in section 4.2.1) was used to calculate sentence embeddings similarities. To capture essential occupational aspects, only core tasks were analysed (definitions in Appendix F.1). Although numerical importance ratings were available, weighting tasks by importance offered no performance gains in earlier research (Dahlke, et. al 2022). Therefore, an unweighted approach was adopted, giving each task equal

weight. Because importance ratings and core/supplementary distinctions were unavailable for DWAs, all DWAs were included equally in the analysis.

To evaluate features against predefined criteria, five key measures were selected based on two factors: (1) the availability of data from O*NET and other sources (2) the use of these criteria in prior O*NET-related research. The selected criteria are **Work Values, Work Styles, Work Interests, Title Similarity, and Salary.**

While Work Values, Styles, and Interests focus on softer, conceptual dimensions of occupational similarity, Title Similarity and Salary emphasize more tangible, practical aspects related to tasks and economic disparity. The table below summarizes these criteria, the basis for their similarity measures, the scales used, methods applied, and example values.

Table Name	Foundation for Similarity Measure	Scale	Example	Method
Work Values	Numerical	Extent (1-7)	Achievement, Data Value: 5.83	Pearson Correlation
Work Styles	Numerical	Importance (1-5)	Initiative, Importance: 3.94	Pearson Correlation
Work Interests	Numerical	Occupational Interest (1-7=	Artistic Data Value: 1.0	Pearson Correlation
Salary	Numerical	Median Salary	Median Salary: $121,200	Absolute Median Difference
Titles	Numerical	Not Applicable	Software Engineer	Cosine Similarity

Table 10: Overview over criteria collected from O*NET: *This table outlines the O*NET data elements considered as criteria for evaluating occupational similarity – work values, styles, interests, salary, and titles - along with their key attributes: table name, foundation for similarity measure, scale (if applicable), method used and example values.*

The similarity criterion calculations were selected based on O*NET's established methods (Dahlke, et. al 2022). Pearson correlation coefficients were used to assess similarities for work values, styles, and interests. Title similarities were calculated using cosine similarity with text embeddings, following the same procedure applied to Activities and Tasks. Salary data from the Bureau of Labor Statistics was merged based on occupation codes, and similarities were calculated using absolute median salary differences to measure disparity directly. Descriptive

statistics and intercorrelations indicate that the measures describe different information about the similarities between the occupations. Descriptive statistics, intercorrelations and information about the specific measures can be found in Appendix F.1.

9.4 Feature Selection

Correlation analysis between criteria and features: To understand how each feature (Tasks, Activities, and KSAs) relates to the occupational similarity criteria (Titles, Values, Styles, Median Salary, and Interests), a correlation analysis was performed. The goal was to determine if feature-based rankings align with criterion-based rankings. For example, if the Title criterion ranks two roles highly similar, the Tasks feature should ideally reflect this similarly.

The analysis (plot in Appendix F.2), reveals that Tasks have the strongest positive correlation with Title Similarity ($r = 0.51$), suggesting that occupations with similar tasks often share similar titles. KSAs are the best predictors for Values Similarity ($r = 0.36$), Styles Similarity ($r = 0.28$), and Interests Similarity ($r = 0.44$), highlighting that foundational competencies and personal attributes are more closely related to these softer dimensions of occupational similarity. Additionally, KSAs display a notable negative correlation with Median Salary Difference ($r = -0.31$), suggesting that greater similarity in knowledge, skills, and abilities is linked to smaller salary disparities between roles. Detailed Work Activities offer a balanced contribution across multiple criteria, with strong correlations to both Title Similarity ($r = 0.45$) and Interests Correlation ($r = 0.41$).

R-squared analysis: Multiple regression analysis was conducted to determine how well different combinations of predictors (KSAs, tasks, DWAs) explained variance in the criteria (values, styles, interests, title, salary). R-squared values were calculated using composite predictors formed by averaging the z-scores of included features.

The regression analysis can be formulated as the general form of linear regression (Montgomery, Peck and Vining 2021):

$$Y = \beta 0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon \quad (9)$$

Where: Y is the dependent variable (criterion: work values, styles, interest, job, title or salary), $X_1, X_2, .., X_n$ are the independent variables (features: KSAs, tasks and DWAs), $\beta 0$ is the intercept, $\beta_1, \ldots, \beta_n$ are the coefficients for each predictor (features) and ϵ is the error term.

	Adjusted R squared				
	Interests Correlation	Median Salary Difference	Styles Correlation	Title Similarity	Values Correlation
Model 1 (KSAs)	0,192	0,097	0,079	0,134	0,130
Model 2 (Tasks)	0,135	0,008	0,043	0,259	0,050
Model 3 (DWAs)	0,166	0,016	0,049	0,202	0,057
Model 4 (Tasks + KSAs)	0,222	0,076	0,085	0,212	0,128
Model 5 (Tasks + DWAs)	0,172	0,013	0,053	0,264	0,061
Model 6 (KSAs + DWAs)	0,230	0,081	0,086	0,191	0,130
Model 7 (Tasks + DWAs + KSAs)	0,238	0,064	0,086	0,241	0,122

Table 11: R squared Analysis for Features and Criteria: *This table shows the adjusted R-squared values for seven models combining different features (KSAs, Tasks, and DWAs) evaluated across five criteria: Interests Correlation, Median Salary Difference, Styles Correlation, Title Similarity, and Values Correlation.*

The analysis highlights that the predictive power of different features varies across dimensions of role similarity. Tasks and Activities (DWAs) provide stronger alignment with Title Similarity due to their direct representation of job functions, while KSAs better capture nuances associated with softer attributes and salary alignment. However, all R^2 values remain relatively low across all criteria, indicating that the true complexity of occupational similarity extends beyond these predictors.

When considering which model to adopt, Model 7 offers the most balanced and comprehensive approach. While the increase in explanatory power over simpler models is often modest, using all three feature helps ensure that no single dimension of similarity is overlooked: KSAs capture underlying competencies, Tasks and DWAs represent the operational aspects of a role, and jointly they account for both the "what" and the "how" of an occupation. Therefore, adopting the full set of features offers a good trade-off between coverage and predictive utility.

It is important to acknowledge potential limitations in this analysis. The use of linear regression assumes independence of observations and a linear relationship between predictors and outcomes (Montgomery, Peck and Vining 2021). In this analysis, each occupation appears in multiple pairs, violating the independence assumption. Despite these limitations, the R-squared

analysis aligns with established practices used in O*NET reports and serves to gain insights into relationships between variables rather than to establish causation.

9.5 Benchmarking Features Against O*NET Rankings

To evaluate the effectiveness of the chosen model 7 (Tasks, DWAs, Knowledge), it was compared with the O*NET dataset on similar occupations. This dataset includes the top 20 most similar roles for each occupation, ranked by an index variable. To quantify the alignment between the rankings, three evaluation metrics where used: Precision@K, Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP). These metrics provide different perspectives on the ranking quality, and all range from 0 to 1, where higher values indicate better performance (Luchen and Clearke 2014). Equations and descriptions of the three measures are provided in Appendix F.3.

The plots below show results for Precision@10, Precision@20, MRR, and MAP. The integrated model, combining Tasks, DWAs, and KSAs, aligns well with the O*NET related occupations benchmark. At K=10, the model achieves a Precision@10 value of 0.65, indicating that 65% of the top 10 recommended occupations appear in O*NET's top 20 similar occupations. When expanding to K=20, the model maintains a Precision@20 of 0.49. An MRR of 0.85 shows relevant occupations are ranked near the top, ensuring quick recommendations. Finally, the MAP score of approximately 0.53 underscores the model's overall quality in ordering relevant occupations higher in the list. These metrics indicate that the model provides a reliable foundation for accurate and effective occupation recommendations, closely reflecting O*NET rankings.

Model	Precision@10	Precision@20	MRR	MAP
Model 7 (Tasks + DWAs + KSAs)	0,648	0,494	0,854	0,533

Table 12: Evaluation of Model 7 Against O*NET Dataset on Occupational Similarity: *This figure shows the evaluation of Model 7 (Tasks, DWAs, and KSAs) using Precision@10, Precision@20, MRR, and MAP metrics. The results highlight the model's alignment with O*NET's occupational rankings.*

9.6 Limitations

The analysis presented above has several limitations. First, the evaluation criteria rely on available O*NET datasets and established methodologies. Choosing features based on their alignment with these criteria is challenging because no perfect criteria exist; if they did, they would already be part of the model's features. Softer measures, such as work styles and values, are inherently abstract and potentially less impactful for determining similarity. Furthermore, the R-squared analysis revealed low overall explained variance, leading to the inclusion of all features to account for different criteria, without assessing the relative importance of each criterion. To address this, involving experts to validate the chosen criteria could enhance the model's credibility and ensure alignment with real-world occupational structures. Additionally, exploring weighted task measures or alternative embedding techniques like GloVe or BERT may improve the robustness and accuracy of the similarity models by identifying more effective methods for capturing occupational similarity.

9.7 Graphical Representation of the Model Output

The final output of the model is visualized through a network graph, where each node represents one of the 878 occupations, and each edge signifies the similarity between roles. By focusing on the top 10 most similar roles for each occupation, we can examine the community structures, the centrality of nodes, and the interconnections between different clusters. The community structure within the graph reveals distinct clusters of occupations that share common attributes in KSAs, tasks, and DWAs. Characteristics of each community and the methodology is described into detail in Appendix F.4. The analysis identified six distinct occupational clusters: administrative, sales, and supervisory roles (Community 0); technical and engineering roles (Community 1), education and healthcare roles (Community 2), manufacturing and machine operation roles (Community 3), manual labour and outdoor roles (Community 4), and IT and data-centric roles (Community 5). Smaller communities, such as Communities 4 and 5, are

more specialized with distinct tasks and skills, while larger communities like Communities 0 and 1 encompass broader occupational categories with varying role similarities. Some overlaps and split clusters, such as those in Community 2 for educational roles, suggest that certain occupations span multiple fields or disciplines. Moreover, the centrality analysis (outlined in Appendix F.4) reveals that managerial and technical occupations hold important positions within the occupational similarity network.

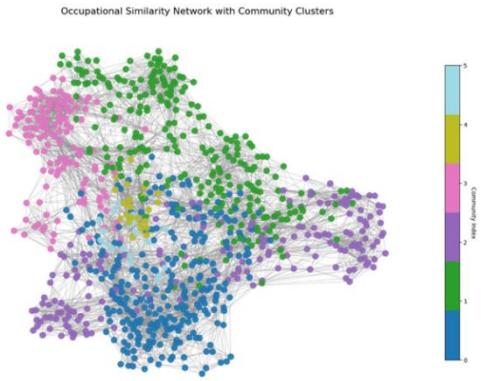
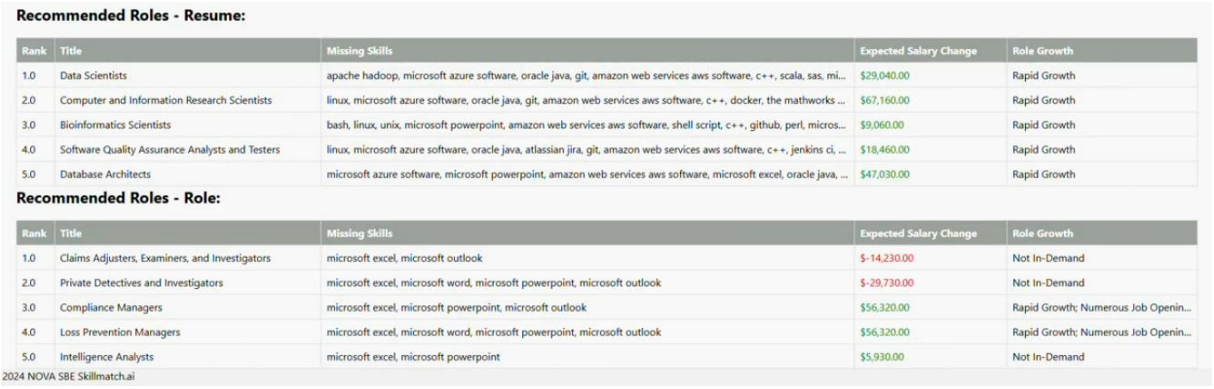


Figure 15: Occupation Cluster Analysis: *This network graph visualizes 878 occupations, where nodes represent roles and edges denote their similarities. The color-coded communities illustrate how occupations group together and connect, reflecting broader occupational categories and specialized fields.*

9.8 Final Output

The final output of the tool integrates résumés matching with a generalized occupational similarity model to deliver role recommendations, displayed in figure 18. The resulting tables include details such as salary information and projected growth for each role in the coming years. By combining individualized skill matching with broader occupational insights, this approach offers a comprehensive solution that considers both personal competencies and role-specific attributes. This model serves as a foundation for developing an internal career development tool aimed at fostering an adaptive and resilient workforce. Detailed code description can be found in Appendix F.4.



Recommended Roles - Resume:

Rank	Title	Missing Skills	Expected Salary Change	Role Growth
1.0	Data Scientists	apache hadoop, microsoft azure software, oracle java, git, amazon web services aws software, c++, scala, sas, mi...	$29,040.00	Rapid Growth
2.0	Computer and Information Research Scientists	linux, microsoft azure software, oracle java, git, amazon web services aws software, c++, docker, the mathworks ...	$67,160.00	Rapid Growth
3.0	Bioinformatics Scientists	bash, linux, unix, microsoft powerpoint, amazon web services aws software, shell script, c++, github, perl, micros...	$9,060.00	Rapid Growth
4.0	Software Quality Assurance Analysts and Testers	linux, microsoft azure software, oracle java, atlassian jira, git, amazon web services aws software, c++, jenkins ci, ...	$18,460.00	Rapid Growth
5.0	Database Architects	microsoft azure software, microsoft powerpoint, amazon web services aws software, microsoft excel, oracle java, ...	$47,030.00	Rapid Growth

Recommended Roles - Role:

Rank	Title	Missing Skills	Expected Salary Change	Role Growth
1.0	Claims Adjusters, Examiners, and Investigators	microsoft excel, microsoft outlook	$-14,230.00	Not In-Demand
2.0	Private Detectives and Investigators	microsoft excel, microsoft word, microsoft powerpoint, microsoft outlook	$-29,730.00	Not In-Demand
3.0	Compliance Managers	microsoft excel, microsoft powerpoint, microsoft outlook	$56,320.00	Rapid Growth; Numerous Job Openin...
4.0	Loss Prevention Managers	microsoft excel, microsoft word, microsoft powerpoint, microsoft outlook	$56,320.00	Rapid Growth; Numerous Job Openin...
5.0	Intelligence Analysts	microsoft excel, microsoft powerpoint	$5,930.00	Not In-Demand

2024 NOVA SBE Skillmatch.ai

Figure 16: Final Output of Résumé-to-Role Ranking Model and Role-to-Role Ranking Model: *The system outputs two tables to cater to different user needs. The first table ranks roles based on cosine similarity derived from the candidate's hard skills, enabling individuals to identify positions that closely align with their existing expertise and experience. The second table allows users to select their current role within the web application and receive a ranked list of similar occupations.*

9.9 Conclusion

This section demonstrated how a role-to-role matching model can address the limitations of résumé-based role recommendations by incorporating broader occupational dimensions such as knowledge, skills, abilities (KSAs), tasks, and detailed work activities (DWAs). The proposed approach moves beyond a narrow focus on technical skills, adopting a skills-first perspective that integrates foundational competencies, and the nature of work performed. Through correlation and R-squared analysis, Model 7, which combines KSAs, tasks, and DWAs, was shown to deliver the most balanced predictions, aligning closely with O*NET benchmarks. The network analysis further revealed clusters of occupations with shared attributes, underscoring the interconnected nature of different roles.

Despite these advancements, certain limitations persist, such as the criteria selection for prediction, low explained variance in R-squared analysis, and limited model testing. These challenges could be mitigated through expert validation, weighted task measures, or alternative embedding techniques. Ultimately, this model lays a foundation for improving role recommendations by combining individual skill sets with a broader understanding of occupational similarities, supporting more accurate and comprehensive career guidance.

10. Building an Ecosystem of Skills-First Applications

In this section we outline the development of Skillmatch.ai. A web application that operationalizes the skills-first approach in the labour market. It integrates functionalities such as résumé screening, tailored interview questions, bespoke apology letters, and role similarity analysis, offering a comprehensive solution for skills-based hiring and career development. Built with a scalable architecture, the application addresses recruitment inefficiencies and skill gaps. Although limitations such as scalability constraints, reliance on external APIs, and the need for real-world data remain, future iterations aim to refine infrastructure, enhance usability, and expand applicability, advancing the adoption of skills-first practices across industries.

10.1 Motivation

The development of the *Skillmatch.ai* web application and its associated skills-first ecosystem is a direct response to the gaps identified in our research on recruitment practices and workforce management as outlined in earlier chapters. Traditional hiring methods, which often prioritize credentials such as degrees and prior job titles, have shown limitations in accurately aligning candidates' abilities with job requirements. These methods frequently overlook individuals with relevant skills but unconventional backgrounds, thereby narrowing the talent pool.

The skills-first approach, which emphasizes competencies over formal qualifications, has emerged as a promising alternative. Despite its potential, the practical implementation of skills-first hiring remains constrained by several factors, including limited access to tools for skill validation, the absence of standardized frameworks, and the organizational resistance associated with adopting new hiring paradigms as outlined in Chapter 3.

To address these challenges, the *Skillmatch.ai* project focuses on operationalizing the skills-first approach by combining advancements in AI and NLP. While the initial model demonstrated potential in automating candidate-job matching and skill evaluation, its accessibility was

restricted to technical users and specific scenarios. This underscored the need for a more user-friendly and scalable interface to enable broader adoption.

The motivation behind developing the *Skillmatch.ai* web application is to provide recruiters and talents with practical tools that facilitate skills-first hiring, including résumé screening, tailored interview questions, bespoke apology letters, and role similarity analysis, all available in a user-friendly application.

This section provides a broad overview of the web application's core components, system architecture, and the key considerations that influenced its design and implementation. It focuses on how the application's functionalities were developed to align with the principles of skills-first hiring.

Given the comprehensiveness of our ecosystem and codebase, we will not be explaining the code in detail here. Rather, relevant code snippets and explanations can be found in Appendix G. Moreover, all our code is provided open source in our GitHub repository. We also made a stable docker image of the web application which can be found on docker hub. Lastly, we also created a demo video of our web application to give the reader an immersive experience of how to interact with the web application.

Link to GitHub: https://github.com/victorbjorsvik/workproject_matching_algo/

Link to Docker: https://hub.docker.com/r/victorbjorsvik/skillmatch_ai

Link to Demo: link

Given the considerations provided above, the following section will serve more as an insight into our web application and provide relevant research, limitations and next steps.

10.2 Core Components

The web application incorporates several core functionalities. The résumé screening functionality incorporates the matching model developed in section 5 and 7 and enables users to input a job description and résumés and receive ranked lists of candidates based on similarity scores. The tailored interview questions endpoint generates interview questions based on the collective pool of skills from the shortlisted candidates and a coding exercise based on the job description. For candidates not selected, the application provides bespoke apology letters, offering professional and empathetic feedback while suggesting resources for skill improvement. Additionally, the role recommendation feature allows job seekers and talents to analyse their skill profiles and identify roles within the database that closely align with their abilities, supporting career exploration and development. A full overview of the user interface file tree can be found in Appendix G.1. In the following section we dive deeper into each of these functionalities.

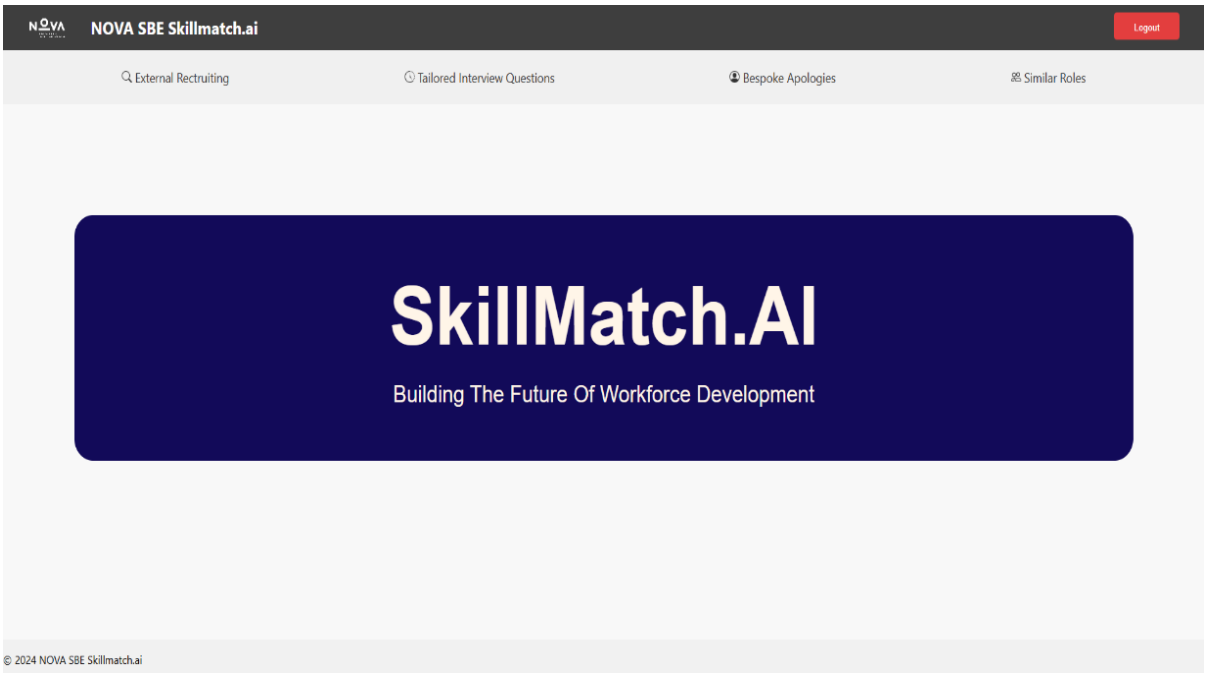


Figure 17: The index page of our web application: *From the navigation bar the user may select each of the four functionalities. The application also has support for several users and an admin user that does not require an API-key for generating the tailored interview questions or bespoke apologies. To interact with the web application, please download and run the docker image provided in the introduction of this chapter.*

10.2.1 Résumé Screening

Résumé screening is a foundational component of the *Skillmatch.ai* web application and serves as the entry point for many of the ecosystem's functionalities. By leveraging the job-resume matching model, developed in section 5 and evaluated and refined in section 7, this feature automates the comparison of job descriptions with candidate résumé to produce ranked similarity scores. These scores not only help recruiters quickly identify top candidates but also provide the basis for other functionalities in the ecosystem, including the tailored interview questions and the bespoke apology letters. This functionality is a core component for operationalizing the skills-first approach, ensuring that candidates are evaluated based on measurable competencies rather than traditional criteria. For pictures of the web application and relevant code snippets for this functionality, see Appendix G.2.

10.2.2 Tailored interview Questions

Traditional job interviews, a cornerstone of the hiring process, often rely on unstructured conversations and subjective evaluations to assess candidates. While interviews can provide valuable insights into a candidate's personality and communication skills, research suggests that traditional interview methods may not be the most effective predictors of their skills and job performance. Moreover, it can introduce biases that hinder fair and objective hiring decision (Dana, Dawes, and Peterson 2013; Judge, Higgins, and Cable 2000; Schmidt and Hunter 1998). Like discussed in section 3, Google implemented their gDroid tool to help mitigate some of these challenges by generating interview questions based solely on the job requirements. Applying the skills-first framework we decided to do something similar, but rather than tailoring the questions to the job requirements we tailor them to the collective pool of the applicants' skills. We also generate a coding exercise based on the job description itself. The questions and exercise were generated using the OpenAI API calling the ChatGPT 4O model. Relevant code can be found in Appendix G.3.

10.2.3 Bespoke Apologies

Rejection is an inevitable part of the job search process. While it can be disheartening for job seekers, it presents an opportunity for employers to leave a positive lasting impression and enhance their employer brand. Traditionally, rejection letters have been short and generic, often leaving candidates feeling undervalued and with no sense of closure (Cortini, Galanti and Barattucci 2019; Jablin and Krone 1984).

We wanted to address these issues by generating more empathetic apology letters that highlight why the applicant in question did not meet the requirements and point the applicant in the direction of relevant resources that might help them improve and expand their skill pool for the next round of applications. See Appendix G.4 for a picture of the functionality and relevant code.

10.2.4 Role Similarity

The role similarity functionality builds on the skills-based similarity model from Chapter 8, enabling job seekers to identify roles that closely align with their skills while highlighting areas for potential upskilling. By leveraging data from the O*NET database and similarity metrics, this feature ranks roles based on skill overlap, providing insights for career transitions and workforce planning. For individuals, this functionality facilitates career development by offering data-driven role recommendations and guidance for targeted upskilling. For organizations, it supports internal mobility by identifying employees suited for in-demand roles, enhancing retention and productivity (Kaminski 2024; European Labour Authority 2023). This feature strengthens the *Skillmatch.ai* ecosystem by advancing the adoption of skills-first practices. Pictures of the functionality and relevant code can be found in appendix G.5.

10.3 System Architecture

The web application was built using the Flask framework, a python framework which is known for its lightweight and modular design, making it a suitable choice for scalable web development (Grinberg 2018). Flask's simplicity allowed us to focus on integrating the job-résumé matching model while keeping the overall architecture maintainable and extensible. The application is structured around a blueprint-based architecture, as seen in the *recruitment* and *roles* directories, which captures related routes and logic. This modular approach enables clear separation of the code, enabling independent development and testing of components (Perras 2015). The **frontend** employs Flask templates located in the *templates* directory, which dynamically render web pages based on user interactions. Static files, such as CSS and images, are stored in the static directory to maintain a clean separation between logic and presentation. The **backend** interacts with the SQLite database *skills.db*, which stores user profiles, job descriptions, and matching scores. Database interactions are managed through helper functions in *helpers.py*. An ER-diagram is provided below.

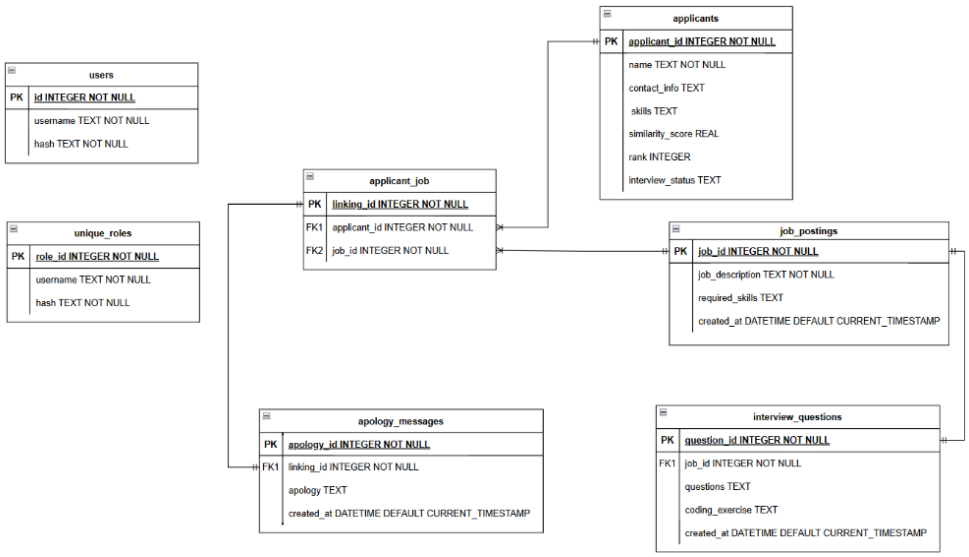


Figure 18: The ER diagram illustrates the database schema for the Skillmatch.ai: *Key tables include users, applicants, job_postings, and interview_questions, with relationships managed through the applicant_job linking table. The schema ensures modularity, allowing efficient storage, querying, and management of user data, job postings, and role-specific features.*

10.4 UI/UX Design

The user interface is designed to be clean and intuitive, with a responsive layout optimized for both desktop and mobile devices. The primary template *layout.html* provides a consistent header, footer, and navigation bar across all pages. Bootstrap was used to ensure responsiveness and pleasing visuals, while custom CSS in *styles.css* provides branding and aesthetic cohesion. Pages such as *index.html* serve as the entry point, while specialized pages like *ext_recuit.html, tailored_interviews.html, bespoke_apology.html,* and *roles.html* focus on the specific features.

10.5 Data Handling and Security

File Handling

Uploaded files are stored in organized subdirectories under *static/uploads*, separating uploads from the matching model *ext/* and the role similarity *roles/*. This structure minimizes file handling errors and is scalable to a certain number of users.

Security

Several security measures were implemented to maintain a secure environment. First, we implemented robust file validation protocols to ensure that only valid PDF files are accepted, thereby preventing malicious uploads and unauthorized access. This was done with custom functions in *helpers.py* and the *werkzeug*-library. Additionally, we employed Flask-Session with filesystem storage to uphold secure, persistent user authentication and session management. Finally, we implemented parameterized queries throughout the application's database interactions to prevent SQL injection attacks and hashed user passwords with the *werkzeug.generate_password_hash* to safeguard sensitive user data.

10.6 Deployment

In this section, we describe the final steps involved in transitioning from a local development environment to a production-ready deployment. During development we used anaconda environments to ensure compatibility across different operating systems. The final web

application was containerized using Docker for reproducible builds and environment consistency. We are not yet hosting the application, but plan on using a cloud platform that provides load balancing and auto-scaling capabilities, ensuring the system can handle increased traffic as adoption grows. Continuous Integration and Continuous Deployment (CI/CD) pipelines will be established to streamline updates and maintenance. After deployment we will utilize monitoring tools like Prometheus or Grafana to proactively address performance bottlenecks and downtime.

10.7 Limitations and Next Steps

While the *Skillmatch.ai* web application demonstrates the potential of a skills-first hiring ecosystem, several limitations must be addressed to improve its scalability, usability, and broader applicability. These limitations primarily pertain to performance constraints, dependency on external tools, and the need for more robust data and infrastructure.

First, due to time constraints the code has a general lack of error handling and testing. This might make the application susceptible to bugs as we start deployment and scaling. The system's reliance on SQLite for database management presents significant scalability challenges. While suitable for the current prototype, SQLite limits the application's ability to handle large datasets or simultaneous user requests. Additionally, the computational demands of the job- résumé matching model hinder its performance when processing extensive candidate pools, especially in real-time scenarios. Transitioning to a more robust database system, such as PostgreSQL, will enhance scalability and support concurrent access by multiple users.

Another limitation arises from the dependency on external APIs, such as OpenAI's GPT models, for generating interview questions and bespoke apologies. These APIs introduce potential latency issues and cost constraints, particularly during periods of high usage. While requiring users to supply their own API keys mitigates cost concerns for individual

organizations, a more integrated and cost-effective approach would improve the system's long-term viability.

The current matching model also suffers from a lack of real-world annotated datasets, which limits the precision of skill extraction and similarity calculations. While synthetic datasets provided an effective means for testing and benchmarking, they do not fully capture the complexities of real-world hiring scenarios. Ethical data collection and partnerships with organizations to access anonymized, high-quality data will be critical for refining the model's accuracy and generalizability.

From a security perspective, the system requires enhanced measures to support enterprise-level deployment. While existing protocols, such as parameterized database queries and file validation, safeguard basic operations, advanced features like multi-factor authentication, role-based access control, and improved session management must be implemented to ensure compliance with data privacy regulations and industry standards.

Future work will focus on addressing these challenges and expanding the system's functionality. First, we will start adding error handling and testing to uncover potential bugs before deployment. Migrating to a more robust database and integrating cloud storage solutions, such as AWS S3, will significantly improve performance and scalability. Usability testing and iterative UI/UX enhancements will ensure the application remains accessible and intuitive for diverse user groups. Additionally, refining API usage, incorporating domain-specific ontologies to improve skill extraction, and extending the system to industries beyond technology will broaden its applicability and impact.

Finally, ethical considerations will remain central to future development, with a focus on reducing biases in the model, ensuring fairness in outcomes, and maintaining strict compliance with data protection regulations. By addressing these limitations, the *Skillmatch.ai* web

application can evolve into a comprehensive, scalable solution for skills-first recruitment and workforce development.

10. 8 Conclusion

The development of the *Skillmatch.ai* web application marks a significant step toward operationalizing the skills-first approach in recruitment and workforce development. By integrating functionalities such as résumé screening, tailored interview questions, bespoke apology letters, and role similarity analysis, the application bridges the gap between theoretical advancements and practical implementation. While the current system demonstrates the feasibility and potential of a skills-first ecosystem, several limitations—ranging from error handling and testing, scalability and performance constraints, reliance on external APIs, and the need for real-world data—highlight areas for further improvement.

Future iterations will focus on enhancing infrastructure, optimizing functionality, and expanding applicability across industries, all while maintaining a strong emphasis on fairness, transparency, and compliance with ethical and data protection standards. Through continuous refinement, the *Skillmatch.ai* ecosystem aims to become a robust, scalable, and accessible solution for modern recruitment challenges, promoting equity and efficiency in the global labour market.

11 Limitations

While this research provided valuable insights into optimizing the Skillmatch.ai pipeline, several limitations must be addressed to ensure its robustness and real-world applicability.

The first major limitation lies in the reliability of the evaluation results, as the model has not yet been tested under real-world hiring conditions. Although the use of synthetic datasets enabled controlled experimentation, the lack of real-world résumés and labeled hiring data prevents us from fully understanding how the model would perform in practical recruitment scenarios. Shifting to real-world data can introduce challenges such as inconsistent skill naming, and unstructured formats. Moreover, testing the model on real data could impact evaluation outcomes and potentially lead to different decisions regarding model design. Capturing real-world résumés with ground-truth hiring labels would significantly enhance the reliability and validity of the current methodology.

Secondly, the model’s reliance on résumés as the primary data input raises conceptual concerns about the effectiveness of a skills-first hiring approach. The developed ecosystem assumes résumés are a sufficient source for extracting candidates' skills for role-to-job matching. However, résumés may not always provide an accurate reflection of candidates' actual competencies, as the skills presented could be overstated, understated, or incomplete. Moreover, the format of résumés imposes inherent limitations, as the one-page structure restricts detail, thereby reducing the model’s ability to extract an accurate skills profile. Objectively measuring skills through alternative, more reliable methods, such as standardized skill assessments, could produce more trustworthy recommendations and strengthen the foundation of a skills-first approach.

Finally, technical limitations hinder the model’s readiness for real-world deployment. The current architecture faces scalability challenges, requiring database management and file handling processes to perform well in production environments.

12 Implications and Future Research

This project employed AI and NLP techniques to develop a skills-first ecosystem that evaluates candidates based on demonstrated abilities rather than traditional credentials. Literature review and data analysis underscore the necessity and potential impact of this approach, confirming that identifying skill overlaps between technical roles enables smoother career transitions. In turn, these findings validate the core premise of skills-first methodologies in talent management.

Beyond theoretical implications, the developed model provides practical applications, such as generating personalized, skill-focused interview questions, automating candidate ranking, and enhancing role-matching. These integrations illustrate how skills-first approaches can improve efficiency in hiring and workforce development. Additionally, the project details the technical processes involved, offering a baseline for future system enhancements.

However, limitations - discussed in detail within the dedicated limitations section - remain. The reliance on synthetic data highlights the need of obtaining larger, more diverse, real-world datasets to strengthen the model's accuracy and generalizability. Issues related to model scalability, as well as the feasibility of a skills-first approach when working with sparse resumes and inconsistently defined skills, pose significant challenges. Addressing these concerns may involve exploring alternative methods and tools to more effectively capture, validate, and integrate candidate skills.

Future research should focus on enlarging the dataset, improving scalability, and extending the model's applicability beyond the technology sector. This expansion includes exploring skill progression tracking, facilitating cross-industry transitions, and mapping meaningful upskilling pathways. Finally, evaluating the long-term market, productivity, and economic outcomes of skills-first practices will be needed to assess their sustainability and impact. By laying this

groundwork, our research contributes to the evolution of skills-first ecosystems, guiding the design of more adaptive, resilient workforce strategies.

13. Conclusion

This project took initial steps toward exploring a skills-first approach in recruitment and workforce development, concentrating on three key areas: identifying skill overlaps to support role transitions, developing a preliminary skills-based model and interface, and navigating the challenges posed by limited data availability.

By examining available datasets (such as Kaggle and O*NET) and applying NLP techniques, including contextual embeddings, we aimed to determine whether skill overlaps could facilitate smoother role transitions. Early findings indicated noteworthy overlaps in skillsets, supporting the hypothesis that skill-oriented mobility is feasible. In parallel, we created a prototype model and a corresponding web application to illustrate how recruiters and job seekers might engage with a skills-focused ecosystem. This included functionalities such as a candidate selection model, role-to-role career guidance, and skill-based interview questions and apology letters, suggesting potential tools for improving hiring practices and enhancing workforce development. Additionally, the use of synthetic data allowed us to refine the model architecture despite data constraints.

Nonetheless, these efforts remain preliminary and limited in scope. The model's reliance on résumés may not fully capture candidates' true competencies, and we have yet to validate its performance with real-world hiring data. Moreover, the current ecosystem is limited to the technology sector and faces issues of scalability. While these results are modest and should be interpreted with caution, we hope they can serve as a starting point to guide further research and development toward more inclusive, adaptable, and skill-focused approaches to talent management and recruitment.

Bibliography

Adewunmi, O. 2024. “The Future of Work: Implications for Global Talent Management.” *IRE Journals* https://www.irejournals.com/formatedpaper/1706289.pdf.

Adomavicius, G., and A. Tuzhilin. 2005. “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.” *IEEE Transactions on Knowledge and Data Engineering*, 734-749, https://doi.org/10.1109/TKDE.2005.99.

Alhadrami, Deema, Till Leopold, Aarushi Singhania, Miral Mir, Bhushan Sethi, and Yalin Zhao. 2023. *Putting Skills First: A Framework for Action.* World Economic Forum.

Allen, Matthew T., Gordon Waugh, Megan Shaw, Suzanne Tsacoumis, David Rivkin, Phil Lewis, Mark Brendle, Denise Craven, Christy Gregory, and Dan Connell. 2012. *Matrix, The Development and Evaluation of a New O*NET® Related Occupations.* Technical report prepared for the National Center for O*NET Development, Human Resources Research Organization, 66 canal center plaza, Suite 700, Alexandria, Virginia 22314-1591: HuMRRO, 54.

Al-Otaibi, Shaha T., & Mourad Ykhlef. 2012. “A survey of job recommender systems.” *International Journal of Physical Sciences*, 5127-5142, https://api.semanticscholar.org/CorpusID:13993376.

Azadeh, A., R. Yazdanparast, S. Abdolhossein Zadeh, and A. Keramati. 2018. “An intelligent algorithm for optimizing emergency department job and patient satisfaction.” *International Journal of Health Care Quality Assurance*, 374-390, https://doi.org/10.1108/IJHCQA-06-2016-0086.

Bankert, Lisa, Nithya Vaduganathan, Sarah Wasserteil, Meera Verghese, Christian Roehri, and Nick Lake. 2024. “Skills-Based Hiring Can Shred the Paper Ceiling.” BCG Global, https://www.bcg.com/capabilities/diversity-inclusion/expert-insights/skill-based-hiring-can-shred-paper-ceiling.

Beyer, Janice M., and Deanna R. Hannah. 2002. “Building on the Past: Enacting Established Personal Identities in a New Work Setting.” *Organization Science* 13 (6): 636-652, https://doi.org/10.1287/orsc.13.6.636.495.

Binvel, Yannick, Michael Franzino, Jean-Marc Laouchez, and Werner Penk. 2018. *Future of Work: The Global Talent Crunch.* Korn Ferry.

Bondarouk, Tanya, Emma Parry, and Elfi Furtmueller. 2016. “Electronic HRM: Four decades of research on adoption and consequences.” *The International Journal of Human Resource Management*, 1-34, https://doi.org/10.1080/09585192.2016.1245672.

Bone, Matthew, Eugenia Ehlinger, and Fabian Stephany. 2024. "Skills or Degree? The Rise of Skill-Based Hiring." *Oxford Internet Institute & Burning Glass Institute* https://doi.org/10.48550/arXiv.2312.11942.

Bowen, David E., Gerald E. Ledford Jr., and Barry R. Nathan. 1991. “Hiring for the Organization, Not the Job.” *Academy of Management Perspectives* 5 (4): 35-51, https://doi.org/10.5465/ame.1991.4274747.

Buchanan, Joy, and Henry Kronk. 2022. “The Slow Adjustment in Tech Labor: Why Do High-Paying Tech Jobs Go Unfilled?” *The Center for Growth and Opportunity at Utah State University* https://www.thecgo.org/research/the-slow-adjustment-in-tech-labor-why-do-high-paying-tech-jobs-go-unfilled/.

Budhwar, Pawan, Ashish Malik, Thedushika De Silva, and Praveena Thevisuthan. 2022. “Artificial intelligence – challenges and opportunities for international HRM: A review and research agenda.” *The International Journal of Human Resource Management*, 1065-1097, https://doi.org/10.1080/09585192.2022.2035161.

Bureau of Labor Statistics. 2023. *May 2023 National Occupational Employment and Wage Estimates.* May. https://www.bls.gov/oes/2023/may/oes_nat.htm.

Caliskan, Aylin, Joanna Bryson, and Arvind Narayanan. 2017. “Semantics derived automatically from language corpora contain human-like biases.” *Science*, 183-186, https://doi.org/10.1126/science.aal4230.

Cantrell, Susan, Michael Griffiths, Robin Jones, and Jeff Hiipakka. 2022. “The Skills-Based Organization: A New Operating Model for Work and the Workforce.” *deloitte.* https://www2.deloitte.com/content/dam/insights/articles/us175310_consulting-the-skills-based-org-report/DI_The-skills-based-organization-report.pdf?trk=public_post_comment-text.

Cappelli, Peter, and J.R Keller. 2014. “Talent Management: Conceptual Approaches and Practical Challenges.” *Annual Review of Organizational Psychology and Organizational Behavior*, 10.1146/annurev-orgpsych-031413-091314.

Chapman, Derek & Webster, Jane. 2003. “The Use of Technologies in the Recruiting, Screening, and Selection Processes for Job Candidates.” *International Journal of Selection and Assessment*, https://doi.org/10.1111/1468-2389.00234.

Chen, Z. 2023. “Collaboration among recruiters and artificial intelligence: Removing human prejudices in employment.” *Cognitive Technology Work*, 135-149, https://doi.org/10.1007/s10111-022-00716-0.

Cheng, Maggie M., and Rick D. Hackett. 2021. “A critical review of algorithms in HRM: Definition, theory, and practice.” *Human Resource Management Review*, January: https://doi.org/10.1016/j.hrmr.2019.100698.

Chicco, Davide, Matthijs J. Warrens, and Giuseppe Jurman. 2021. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation." *PeerJ Computer Science 7:e623*, June 6: 1-4, 10.7717/peerj-cs.623.

Chowdhury, Gobinda. 2005. “Natural language processing.” *ARIST*, 51-89, https://doi.org/10.1002/aris.1440370103.

Cohen, I., Y. Huang, J. Chen, and J. Benesty. 2009. “Pearson Correlation Coefficient.” *Noise reduction in speech processing,* , 1-4, DOI 10.1007/978-3-642-00296-0_5.

Cortini, Michela, Teresa Galanti, and Massimiliano Barattucci. 2019. “The Effect of Different Rejection Letters on Applicants’ Reactions.” *Behavioral Sciences* https://doi.org/10.3390/bs9100102.

Council, European Parliament and. 2024. “Official Journal of the European Union.” *Regulation (EU) 2024/1689 of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence and Amending Certain Union Legislative Acts (Artificial Intelligence Act).* https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

Dahlke, Jeffrey A., Dan J. Putka, Ori Shewach, and Phil Lewis. 2022. *Developing Related Occupations for the O*NET Program.* Technical report prepared for the National Center for O*NET Development, 66 Canal Center Plaza, Suite 700, Alexandria, VA 22314: The Human Resources Research Organization (HumRRO) , 40.

Dana, Jason, Robyn Dawes, and Nathanial Peterson. 2013. “Belief in the unstructured interview: The persistence of an illusion.” *Judgment and Decision Making* 512-520.

De Vos, Simon, Johannes De Smedt, Marijke Verbruggen, and Wouter Verbeke. 2024. “Data-driven internal mobility: Similarity regularization gets the job done.” *Knowledge-Based Systems*, https://doi.org/10.1016/j.knosys.2024.111824.

Delecraz, Sebastien, Loukman Eltarr, Martin Becuwe, Henri Bouxin, Nicolas Boutin, and Olivier Oullier. 2022. “Responsible Artificial Intelligence in Human Resources Technology: An innovative inclusive and fair by design matching algorithm for job recruitment purposes.” *Journal of Responsible Technology*, https://doi.org/10.1016/j.jrt.2022.100041.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *Association for Computational Linguistics* 4171–4186.

Di Battista, Attilio, Sam Grayling, Elselot Hasselaar, Till Leopold, Ricky Li, Mark Rayner, and Saadia Zahidi. 2023. *Future of Jobs Report.* https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf: World Economic Forum. https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf.

Dokko, Gina, Steffanie Wilk, and Nancy Rothbard. 2009. “Unpacking prior experience: How career history affects job performance.” *Organization Science* 20 (1): 51-68, https://doi.org/10.1287/orsc.1080.0357.

Dokmanic, Ivan, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. 2015. "Euclidean Distance Matrices: Essential theory, algorithms, and applications." *IEEE Signal Processing Magazine*, November: 10.1109/MSP.2015.2398954.

Dridri, Amira. 2022. *Job-Resume-Matching.* https://github.com/amiradridi/Job-Resume-Matching.

Durth, Sandra, Asmus Komm, Florian Pollner, and Angelika Reich. 2023. “Reimagining People Development to Overcome Talent Challenges.” McKinsey & Company, https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/reimagining-people-development-to-overcome-talent-challenges.

Emmons, Scott, Stephen Kobourov, Mike Gallant1, and Katy Börner. 2016. “Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale.” *PLoS ONE 11 (7)*, 8 July: 18, 10.1371/journal.pone.0159161.

Esch, P. van, J. S. Black, and J. Ferolie. 2019. “Marketing AI recruitment: The next phase in job application and selection.” *Computers In Human Behavior*, 215-222, https://doi.org/10.1016/j.chb.2018.09.009.

European Labour Authority. 2023. *Five ways AI can help your job search.* December. https://eures.europa.eu/five-ways-ai-can-help-your-job-search-2023-12-19_en.

Filho, Othamar Gama. 2020. “Chief Executive.” *Artificial Intelligence Tech Is Evolving The Recruiting Process.* 17 September. https://chiefexecutive.net/artificial-intelligence-tech-evolving-recruiting-process/.

Fraij, J., and V. László. 2021. “A Literature Review: Artificial Intelligence Impact on the Recruitment Process.” *International Journal of Engineering and Management Sciences*, 108-119, https://doi.org/10.21791/IJEMS.2021.1.10.

Françoise, Delamare Le Deist, and Jonathan Winterton. 2005. "What Is Competence?" *Human Resource Development International, Volume 8, 2005 - Issue 1*, March: 27-46, https://doi.org/10.1080/1367886042000338227.

Franklin, James Belanich. 2016. “Review and Assessment of Personnel Competencies and Job Description Models and Methods.” *Institute for Defense Analyses* Review and Assessment of Personnel Competencies and Job Description Models and Methods.

Freire, Mauricio, and Leandro de Castro. 2021. “e-Recruitment recommender systems: a systematic review.” *Knowledge and Information Systems*, 1-20, https://doi.org/10.1007/s10115-020-01522-8.

Fuller, Joseph, Christina Langer, and Matthew Sigelman. 2022. “Skills-Based Hiring Is on the Rise.” *Harvard Business Review* https://hbr.org/2022/02/skills-based-hiring-is-on-the-rise.

Fuller, Joseph, Manjari Raman, Eva Sage-Gavin, and Kristen Hines. 2021. "Hidden Workers: Untapped Talent." *Harvard Business Review* (Harvard Business School) https://www.hbs.edu/managing-the-future-of-work/Documents/research/hiddenworkers09032021.pdf.

Gale, D., and L. S. Shapley. 1962. “College Admissions and the Stability of Marriage.” *The American Mathematical Monthly*, 9-15, https://doi.org/10.1080/00029890.1962.11989827.

Garg, S., S. Sinha, A.K. Kar, and M. Mani. 2022. “A review of machine learning applications in human resource management.” *International Journal of Productivity and Performance Management*, 1590-1610, https://doi.org/10.1108/IJPPM-08-2020-0427.

Gilardi, F., M. Alizadeh, and M. Kubli. 2023. “ChatGPT outperforms crowd workers for text-annotation tasks.” *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, https://doi.org/10.1073/pnas.2305016120.

Goutte, Cyril, and Éric Gaussier. 2005. *A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation.* Springer Berlin Heidelberg., Berlin Heidelberg.: Springer, https://doi.org/10.1007/978-3-540-31865-1_25.

Grinberg, Miguel. 2018. “Flask Web Development: Developing Web Applications with Python 2nd edition.” 10. Sebastopol: O'Reilly Media Inc.

Gusdorf, Myrna L. 2008. “Recruitment and Selection: Hiring the Right Person.” *Society for Human Resource Management* https://www.shrm.org/content/dam/en/shrm/credentials/shrm-certification/teaching-resources/recruitment-and-selection-hiring-the-right-person-instructors-manual.pdf.

Hadzilacos, Rigas, Till Alexander Leopold, Vesselina Ratcheva, Saadia Zahidi, Thomas Fink, Yannick Lingelbach, Rainer Strack, et al. 2019. *Towards a Reskilling Revolution* . https://www3.weforum.org/docs/WEF_Towards_a_Reskilling_Revolution.pdf, Centre for New Economy and Society Insight Report, The World Economic Forum, 93, https://www3.weforum.org/docs/WEF_Towards_a_Reskilling_Revolution.pdf.

Han, Jiawei, Micheline Kamber, and Jian Pei. 2012. “2 - Getting to Know Your Data.” In *Data Mining (Third Edition)*, by Jiawei Han, Micheline Kamber and Jian Pei, 77-80.

Hancock, Bryan, Chris Higgins, Jordan Law, Sarah Olson, Neel Patel, and Katie Van Dusen. 2022. “Taking a Skills-Based Approach to Building the Future Workforce.” *McKinsey & Company* https://www.mckinsey.com/capabilities/people-and-organizational-

performance/our-insights/taking-a-skills-based-approach-to-building-the-future-workforce.

Hao, Shuang, Wenfeng Han, Tao Jiang, Yiping Li, Haonan Wu, Chunlin Zhong, Zhangjun Zhou, and He Tang. 2024. “Synthetic Data in AI: Challenges, Applications, and Ethical Implications.” *arXiv*, https://doi.org/10.48550/arXiv.2401.01629.

He, Haibo, and E.A. Garcia. 2009. “Learning from Imbalanced Data.” *IEEE Transactions on Knowledge and Data Engineering*, 1263–1284, https://doi.org/10.1109/TKDE.2008.239.

Heidarian, Arash, and Michael J. Dinneen. 2016. "A Hybrid Geometric Approach for Measuring Similarity Level Among Documents and Document Clustering." *Second International Conference on Big Data Computing Service and Applications978-1-5090-2251-9/16 $31.00 © 2016 IEEEDOI 10.1109/BigDataService.2016.14142Authorized licensed use limited to: b-on: Universidade de Lisboa Reitoria. Downloaded on December 10,*, March 1: 10.

Hewage, Anusha. 2023. “Exploring the Applicability of Artificial Intelligence in Recruitment and Selection Processes: A Focus on the Recruitment Phase.” *Journal of Human Resource and Sustainability Studies*, 603-634, https://doi.org/10.4236/jhrss.2023.113034.

Hocken, Emily, and Grace King. 2023. “AI in the hiring process.” *Strategic HR Review*, 81-84, https://doi.org/10.1108/SHR-03-2023-0014.

Holm, A. B. 2012. “E-recruitment: Towards an Ubiquitous Recruitment Process and Candidate Relationship Management.” *German Journal of Human Resource Management*, 241-259, https://doi.org/10.1177/239700221202600303.

Honnibal, Matthew, and Ines Montani. 2017. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.” *To appear.*

Hossain, Ekram. 2024. *Machine Learning Crash Course for Engineers.* Springer.

Huggingface. 2024. *Sentence-Transformers.* Accessed 12 10, 2024. https://huggingface.co/sentence-transformers?sort_models=downloads#models.

Hunkenschroer, A.L., and C. Luetge. 2022. "Ethics of AI-Enabled Recruiting and Selection: A Review and Research Agenda." *Journal of Business Ethics*, 977-1007, https://doi.org/10.1007/s10551-022-05049-6.

Hunter, John E., and Ronda F. Hunter. 1984. “Validity and Utility of Alternative Predictors of Job Performance.” *Psychological Bulletin* 96 (1): 72–98, https://doi.org/10.1037/0033-2909.96.1.72.

Jablin, Fredrick M., and Kathleen Krone. 1984. “Characteristicsof Rejection Letters and their Effects on Job Applicants.” *Written Communication* 387-406.

Jordon, James, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N. Cohen, and Adrian Weller. 2022. “Synthetic Data – What, Why and How?” *arXiv*, https://doi.org/10.48550/arXiv.2205.03257.

Judge, Timothy A., Chad A. Higgins, and Daniel M. Cable. 2000. “THE EMPLOYMENT INTERVIEW: A REVIEW OF RECENT RESEACRH AND RECOMMENDATIONS FOR FUTURE RESEARCH.” *Human Resource Management Review* 383-406.

Kaminski, Adrian. 2024. “Leveraging Artificial Intelligence in Human Resource Management: A Study of Adoption and Impact in the Australian Commercial Media Industry.” 18, https://doi.org/10.20944/preprints202409.1517.v1).

Kara, A., F.S. Daniş, G.K. Orman, S.N. Turhan, and Ö.A. Özlü. 2023. “Job Recommendation Based on Extracted Skill Embeddings.” In *Intelligent Systems and Applications*, by Kohei Arai, https://doi.org/10.1007/978-3-031-16075-2_35. Cham, Switzerland: Springer.

Kavanagh, M. L., H. G. Gueutal, and S. I. Tannenbaum. 1990. *Human Resource Information Systems: Development and Application.* PWS-Kent Publishing Company.

Kiss, Andrea, Robert Garlick, Kate Orkin, and Lukas Hensel. 2023. “Jobseekers’ Beliefs about Comparative Advantage and (Mis)Directed Search.” *Upjohn Institute* https://docs.iza.org/dp16522.pdf.

Konstantinidis, Ioannis, Manolis Maragoudakis, Ioannis Magnisalis, Christos Berberidis, and Vassilios Peristeras. 2022. “Knowledge-driven Unsupervised Skills Extraction for Graph-based Talent Matching.” *Proceedings of the ACM International Conference on Information and Knowledge Management.* 1-7, https://doi.org/10.1145/3549737.3549769.

Kristof, Amy L. 1996. “Person–Organization Fit: An Integrative Review of Its Conceptualizations, Measurement, and Implications.” *Personnel Psychology* 49 (1): 1-49, https://doi.org/10.1111/j.1744-6570.1996.tb01790.x.

Kumar, Vijay, Jitender Chhabra, and Dinesh Kumar. 2013. “Impact of Distance Measures on the Performance of Clustering Algorithms.” *International Conference on Advanced Computing, Networking, and Informatics*, 7.

Kużelewska, Urszula. 2014. “Clustering Algorithms in Hybrid Recommender System on MovieLens Data.” *Studies in Logic, Grammar and Rhetoric* , 1 January: 15.

Lavelli, A., Califf, M. E., Ciravegna, F., Freitag, D., Giuliano, C., Kushmerick, N., & Romano, L. 2004. “A Critical Survey of the Methodology for IE Evaluation.” *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC’04).* Lisbon, Portugal: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2004/pdf/416.pdf.

Lee, Chankyu, Roy Rajarshi, Xu Mengyao, Raiman Jonathan, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. "NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models ." https://doi.org/10.48550/arXiv.2405.17428.

Liashchynskyi, Petro, and Pavlo Liashchynskyi. 2019. “Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS.” *arXiv, Cornell University* 11, https://arxiv.org/abs/1912.06059.

LinkedIn. 2023. “Skills-First: Reimagining the Labor Market and Breaking Down Barriers.” *Linkedin Economic Graph.* https://economicgraph.linkedin.com/research/skills-first-report.

Little, Roderick J.A. 1993. “Statistical Analysis of Masked Data.” *Journal of Official Statistics*, 407.

Liu, Ruibo, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, et al. 2024. “Best Practices and Lessons Learned on Synthetic Data.” https://doi.org/10.48550/arXiv.2404.07503.

Luchen, Tan, and Charles L.A Clearke. 2014. "A Family of Rank Similarity Measures Based on Maximized Effectiveness Difference." *IEEE Transactions on Knowledge and Data Engineering*, August 14: 13, 10.1109/TKDE.2015.2448541.

Magnet, Sophie. 2024. *365datascience.* May 22. https://365datascience.com/career-advice/career-guides/machine-learning-engineer-skills/.

Magron, A., A. Dai, M. Zhang, S. Montariol, and A. Bosselut. 2024. “JOBSKAPE: A Framework for Generating Synthetic Job Postings to Enhance Skill Matching.” *arXiv*, https://doi.org/10.48550/arXiv.2402.03242.

Manoharan, T. R., C. Muralidharan, and S. G. Deshmukh. 2011. “An integrated fuzzy multi-attribute decision-making model for employees’ performance appraisal.” *The International Journal of Human Resource Management*, 722-745, https://doi.org/10.1080/09585192.2011.543763.

Marler, Janet, and Sandra Fisher. 2013. “An evidence-based review of E-HRM and strategic human resource management.” *Human Resource Management Review*, 18-36, https://doi.org/10.1016/j.hrmr.2012.06.002.

Martinez-Gil, Jorge, Alejandra Lorena Paoletti, and Mario Pichler. 2020. “A Novel Approach for Learning How to Automatically Match Job Offers and Candidate Profiles.” *Information Systems Frontiers*, December: 1265-1274, https://doi.org/10.1007/s10796-019-09929-7.

McDaniel, M. A., F. L. Schmidt, and J. E. Hunter. 1988. “Job experience correlates of job performance.” *Journal of applied psychology* 73 (2): 327, https://doi.org/10.1037/0021-9010.73.2.327.

McHugh, Mary L. 2013. “The Chi-square test of independence.” *Lessons in biostatistics*, 1 April: 7.

McRae, Elizabeth R., Patrick Aykens, Kelsey Lowmaster, and Jennifer Shepp. 2024. “9 Trends That Will Shape Work in 2024 and Beyond.” *Harvard Business Review* https://hbr.org/2024/01/9-trends-that-will-shape-work-in-2024-and-beyond.

Meng, Rui, Liu Ye, Rayhan, Joty Shafiq, Xiong Caiming, Zhou Yingbo, and Yavuz Semih. 2024. “SFR-Embedding-Mistral: Enhance Text Retrieval with Transfer Learning.” *Salesforce AI Research Blog.* https://blog.salesforceairesearch.com/sfr-embedded-mistral/.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Distributed representations of words and phrases and their compositionality.” *Advances in Neural Information Processing Systems* b.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” a.

Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. 2021. *Introduction to Linear Regression Analysis.* Vol. 5th ed. Hoboken, NJ: John Wiley & Sons.

Mooney, Paul. 2022. *Kaggle Machine Learning & Data Science Survey.* https://kaggle.com/competitions/kaggle-survey-2022.

Moreira, Gabriel, Osmulski Radek, Xu Mengyao, Ak Ronay, Schifferer Benedikt, and Oldridge Even. 2024. "NV-Retriever: Improving text embedding models with effective hard-negative mining." https://doi.org/10.48550/arXiv.2407.15831.

Moritz, Robert E., and Saadia Zahidi. 2023. “Putting Skills First: A Framework for Action.” *World Economic Forum.*

Muennighoff, Niklas, Nabila Tazi, Louis Magne, and Nils Reimers. 2023b. *MTEB: Massive Text Embedding Benchmark.* arXiv Preprint., https://doi.org/10.48550/arXiv.2210.07316.

Mustafaa, B., and S Lleshi. 2024. “The impact of lifelong learning and investments in employee development on employee productivity and performance.” *Multidisciplinary Reviews*, https://doi.org/10.31893/multirev.2024175.

National Center for ONET Development. 2024. “Task Statements.” *ONET Resource Center* . 19 November. https://www.onetcenter.org/dictionary/29.1/excel/task_statements.html.

NetworkX Developers. 2024. *NetworkX 3.4.2 Documentation.* https://networkx.org/documentation/stable/reference/algorithms/community.html.

Niehueser, W., and G. Boak. 2020. “Introducing artificial intelligence into a human resources function.” *Industrial and Commercial Training*, 121-130, https://doi.org/10.1108/ICT-10-2019-0097.

O*NET Resource Center. 2024. *onetcenter.* November 19. Accessed Oktober 1, 2024. https://www.onetcenter.org/dictionary/29.1/excel/scales_reference.html.

OpenAI. 2024. *GPT-4o Technical Documentation.* https://platform.openai.com/docs/models/gpt-4o.

—. 2024. *System and User Messages in API Interactions.* https://platform.openai.com/docs/api-reference/chat/create.

Otani, Naoya, Yosuke Otsubo, Tetsuya Koike, and Masashi Sugiyama. 2020. “Binary Classification With Ambiguous Training Data.” *Machine Learning*, 2369–2388, https://doi.org/10.1007/s10994-020-05915-2.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. “GloVe: Global Vectors for Word Representation.” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP.* Doha: Association for Computational Linguistics. 1532–1543.

Perras, Joel. 2015. “Flask Blueprints.” 18-21. Birmingham: Packt Publishing.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. “Deep contextualized word representations.”

Petrican, Teodor, Ciprian Stan, Marcel Antal, Ioan Salomie, Tudor Cioara, and Ionut Anghel. 2017. “Ontology-based Skill Matching Algorithms.” *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP).* 205-211, https://doi.org/10.1109/ICCP.2017.8117005.

Plackett, R. L. 1983. “Karl Pearson and the Chi-Squared Test .” *International Statistical Institute (ISI)*, https://doi.org/10.2307/1402731.

Powers, David Martin. 2020. "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation." https://doi.org/10.48550/arXiv.2010.16061.

Pundir, Rahul, Anil Dhasmana, Urmila Karakoti, Aishki Sikder, Shreya Sharma, and Mahesh Manchanda. 2024. “Enhancing Resume Recommendation System through Skill-based Similarity using Deep Learning Models.” *IEEE International Conference on Information and Communication Technology (ICICT).* 557-562, https://doi.org/10.1109/ICICT60155.2024.10544875.

Raghavan, Manish, Solon Barocas, Jon Kleinberg, and Karen Levy. 2020. “Mitigating bias in algorithmic hiring: Evaluating claims and practices.” *2020 Conference on Fairness,*

Accountability, and Transparency (FAT '20).* New York, NY, USA: Association for Computing Machinery (ACM). 469-481, https://doi.org/10.1145/3351095.3372828.

Rajput, D., W.J. Wang, and C.C. Chen. 2023. "Evaluation of a Decided Sample Size in Machine Learning Applications." *BMC Bioinformatics*, https://doi.org/10.1186/s12859-023-05156-9.

Raschka, Sebastian. 2018. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning." http://doi.org/10.48550/arXiv.1811.12808.

Ratner, Alexander, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. "Data Programming: Creating Large Training Sets, Quickly." *Advances in Neural Information Processing Systems.* 3567–3575, https://doi.org/10.48550/arXiv.1605.07723.

Reimers, Nils, and Iryna Gurevych. 2019. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." 2.

Ren, Jing, Feng Xia, Xiangtai Chen, Jiaying Liu, Mingliang Hou, Ahsan Shehzad, Nargiz Sultanova, and Xiangjie Kong. 2021. "Matching Algorithms: Fundamentals, Applications and Challenges." *IEEE Transactions on Emerging Topics in Computational Intelligence*, 332-350, https://doi.org/10.1109/TETCI.2021.3067655.

Resnick, P., and H. R. Varian. 1997. "Recommender Systems." *Communications of the ACM*, 56-58, https://doi.org/10.1145/245108.245121.

Ricci, F., L. Rokach, and B. Shapira. 2021. "Recommender Systems: Techniques, Applications, and Challenges." In *Recommender Systems Handbook*, by F. Ricci, L. Rokach and B. Shapira, 1-35, https://doi.org/10.1007/978-1-0716-2197-4_1. Cham, Switzerland: Springer eBooks.

Rojas-Galeano, Sergio, Posada P., Jorge E., & Ordoñez, Esteban. 2022. "A Bibliometric Perspective on AI Research for Job-Résumé Matching." *The Scientific World Journal*, 1-15, https://doi.org/10.1155/2022/8002363.

Roth, Alvin E., and Marilda A. Oliveira Sotomayor. 1990. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis.* Cambridge: Cambridge University Press.

Rubin, Donald B. 1993. "Discussion: Statistical Disclosure Limitation." *Journal of Official Statistics*, 461-468.

—. 1987. *Multiple Imputation for Nonresponse in Surveys.* New York: John Wiley & Sons.

Saad, Zuria Akmal, Muhammad Ashraf Fauri, Naliza Solat, and Azmi Salim. 2023. "Measuring Signal of Skill Mismatch from Online Job Advertisements and HR Roles: A conceptual paper." *Environment-Behaviour Proceedings Journal* 293–300, https://doi.org/10.21834/e-bpj.v8iSI15.5071.

Schmidt, Frank L., and J.E. Hunter. 1998. "The Validity and Utility of Selection Methods in Personnel Psychology: Practical and Theoretical Implications of 85 Years of Research Findings." *Psychological Bulletin* 262-274.

Schober, Patric. 2018. "Correlation Coefficients: Appropriate Use and Interpretation." *Anesthesia & Analgesia*, May: 1763-1768, 10.1213/ANE.0000000000002864.

Schröer, Christoph, Felix Kruse, and Jorge Marx Gómez. 2021. "A Systematic Literature Review on Applying CRISP-DM Process Model." *Procedia Computer Science* 181: 526–534.: https://doi.org/10.1016/j.procs.2021.01.199.

Sebastian, Raschka. 2018. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning." http://doi.org/10.48550/arXiv.1811.12808.

Sedgwick, Philip. 2012. "Pearson's correlation coefficien." https://doi.org/10.1136/bmj.e4483.

Shakya, Aman, and Subhash Paudel. 2019. "Job-Candidate Matching using ESCO Ontology." *Journal of the Institute of Engineering* 1-13.

Sigelman, Matthew, Joseph Fuller, and Alex Martin. 2024. "Skill-Based Hiring: The Long Road from Pronouncements to Practice." *Harvard Business Review* https://www.voced.edu.au/content/ngv:100678.

Singla, A. 2023. “Machine Learning Operations (MLOps): Challenges and Strategies.” *Journal of Knowledge Learning and Science Technology*, 333-340, https://doi.org/10.60087/jklst.vol2.n3.p340.

Skondras, Panagiotis, Panagiotis Zervas, and Giannis Tzimas. 2023. “Generating Synthetic Resume Data with Large Language Models for Enhanced Job Description Classification.” *MDPI*, 1-12, https://doi.org/10.3390/fi15110363.

Sohangir, Sahar, and Wang Dingding. 2017. "Improved sqrt-cosine similarity measurement." *Journal of Big Data*, July 25: 13.

Song, Kaitao, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. “MPNet: Masked and Permuted Pre-training for.” *34th Conference on Neural Information Processing Systems (NeurIPS 2020).* Vancouver.

Speranskaya, Marina, Martin Schmitt, and Benjamin Roth. 2021. “Ranking vs. Classifying: Measuring Knowledge Base Completion Quality.” https://doi.org/10.48550/arXiv.2102.06145.

Steck, Harald. 2013. “Evaluation of Recommendations: Rating-Prediction and Ranking.” In *RecSys '13: Proceedings of the 7th ACM conference on Recommender systems*, by Qiang Yang, Irwin King and Qing Li, 213 - 220, https://doi.org/10.1145/2507157.250716. New York: Association for Computing Machinery.

Steck, Harald, Chaitanya Ekanadham, and Nathan Kallus. 2024. “Is Cosine-Similarity of Embeddings Really About Similarity?” *Netflix Inc.* 4.

Strohmeier, Stefan. 2007. “Research in e-HRM: Review and implications.” *Human Resource Management Review*, 19-37, https://doi.org/10.1016/j.hrmr.2006.11.002.

Stybel, Laurence J. 2010. “Managing the Inner Contradictions of Job Descriptions: A Technique for Use in Recruitment.” *The Psychologist-Manager Journal* 10.1080/10887151003776554.

Subramaniam, Jayaraman, Parvaiz Talib, and Ahmad Faraz Khan. 2018. “Integrated Talent Management Scale: Construction and Initial Validation.” *SAGE Open*, 10.1177/2158244018780965.

Tambe, P., P. Cappelli, and V. Yakubovich. 2019. “Artificial Intelligence in Human Resources Management: Challenges and a Path Forward.” *California Management Review*, 15-42, https://doi.org/10.1177/0008125619867910.

Tannock, Stuart. 2008. “The Problem of Education-Based Discrimination.” *British Journal of Sociology of Education* 29 (5): 439–449, https://doi.org/10.1080/01425690802326846.

Tejaswini, K., V. Umadevi, Shashank Kadiwal, and Sanjay Revanna. 2021. “Design and Development of Machine Learning Based Resume Ranking System.” *Global Transitions Proceedings*, https://doi.org/10.1016/j.gltp.2021.10.002.

The White House. 2022. *Fact Sheet: Background on President Biden’s Remarks on the Economy During His First State of the Union Address.* The White House.

Thoma, Martin, mara004, Kyungwon Chun, and Marek Kubowicz. 2023. *PDF Library Benchmarks.* https://github.com/py-pdf/benchmarks?tab=readme-ov-file#pdf-library-benchmarks.

Valente, Thomas W, Kathryn Coronges, Cynthia Lakon, and Elizabeth Costenbader. 2008. “How Correlated Are Network Centrality Measures?” *Connect (Tor)*, January: 16-26, 10.1007/s11121-007-0072-0.

Van Iddekinge, Chad H., John D. Arnold, Rachel E. Frieder, and Philip L. Roth. 2019. “A Meta-Analysis of the Criterion-Related Validity of Prehire Work Experience.” *Personnel Psychology* 72 (4): 571-598, https://doi.org/10.1111/peps.12335.

Van-Duyet, Le, Vo Minh Quan, and Dang Quang An. 2017. “Skill2vec: Machine Learning Approach for Determining the Relevant Skills from Job Description.”

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gome, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” https://doi.org/10.48550/arXiv.1706.03762.

Voigt, Paul, and Axel Bussche. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide.* Springer.

Wang, Wenhui, Wei Furu, Dong Li, Bao Hangbo, Yang Nan, and Zhou Ming. 2020. "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers." (Advances in Neural Information Processing Systems 33: 5776–5788.) https://doi.org/10.48550/arXiv.2012.15828.

Wilk, Steffanie L., and Peter Cappelli. 2006. "Understanding the Determinants of Employer Use of Selection Methods." *Personnel Psychology* 56 (1): 103–124, https://doi.org/10.1111/j.1744-6570.2003.tb00145.x.

Wirth, Rüdiger, and Jochen Hipp. 2000. "CRISP-DM: Towards a Standard Process Model for Data Mining." 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, 29–39. Manchester, UK, https://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” https://doi.org/10.48550/arXiv.1906.08237.

Zhang, H. 2024. “Exploring the Impact of AI on Human Resource Management: A Case Study of Organizational Adaptation and Employee Dynamics.” *IEEE Transactions on Engineering Management*, https://doi.org/10.1109/TEM.2024.3457520.

Zhao, Jing, Jingya Wang, Madhav Sigdel, Bopeng Zhang, Phuong Hoang, Mengshu Liu, and Mohammed Korayem. 2021. “Embedding-based Recommender System for Job to Candidate Matching on Scale.” *Preprint*, https://doi.org/10.48550/arXiv.2107.00221.

Appendix A - EDA

This appendix supports the group portion of the project, where each paragraph is aligned with the corresponding headings in the main document. It covers various stages of data preparation, including cleaning, curation, and feature selection from Chapter 4. Additionally, the appendix outlines the methodologies used for similarity calculation and includes industry distribution insights. Additionally, the codebase structure and detailed Python snippets for skill extraction and analysis is included.

A.1 Skill Exploratory Data Analysis

A.1.1 Data Cleaning & Curation

Role Cleaning: 13 367 respondents did not specify a role and were excluded from the analysis. The survey included roles such as Data Scientist, Data Analyst (Business, Marketing, Financial, Quantitative, etc.), Currently Not Employed, Software Engineer, Teacher/Professor, Manager (Program, Project, Operations, Executive-level, etc.), Research Scientist, ML/MLOps Engineer, Engineer (Non-software), Data Engineer, Statistician, Data Architect, Data Administrator, Developer Advocate, and Other.

The distribution of these roles is given in the plot below.

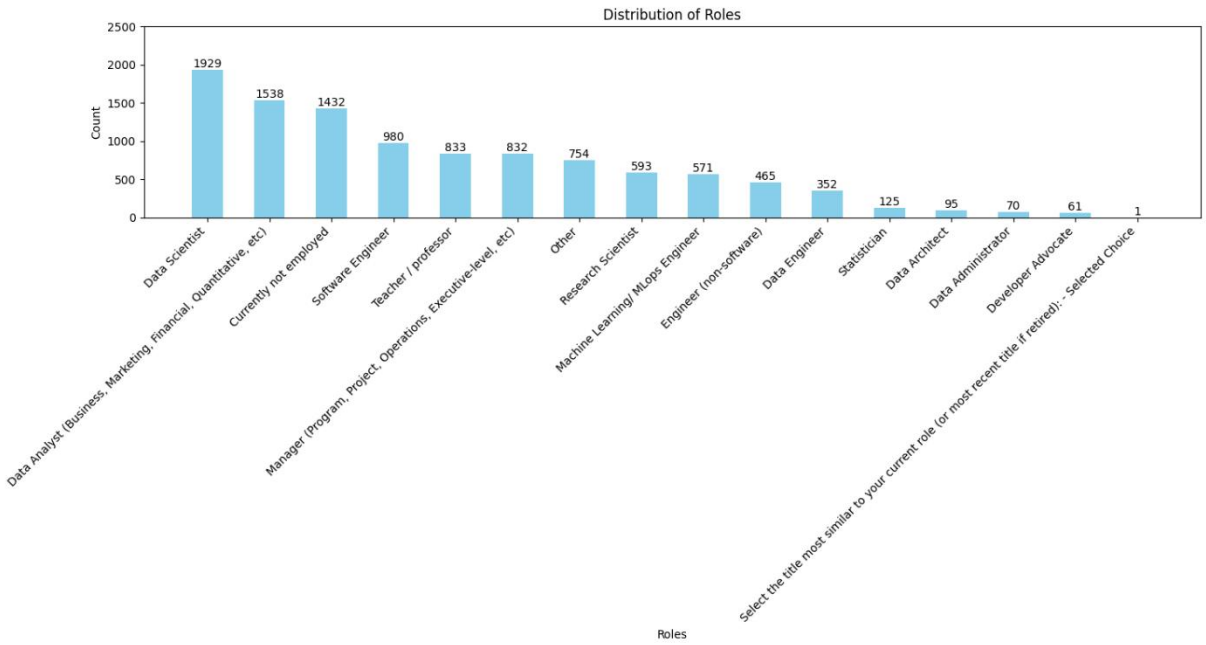


Figure 19: *Distribution of Roles Before Data Cleaning*

After cleaning, the distribution of Roles is as follows:

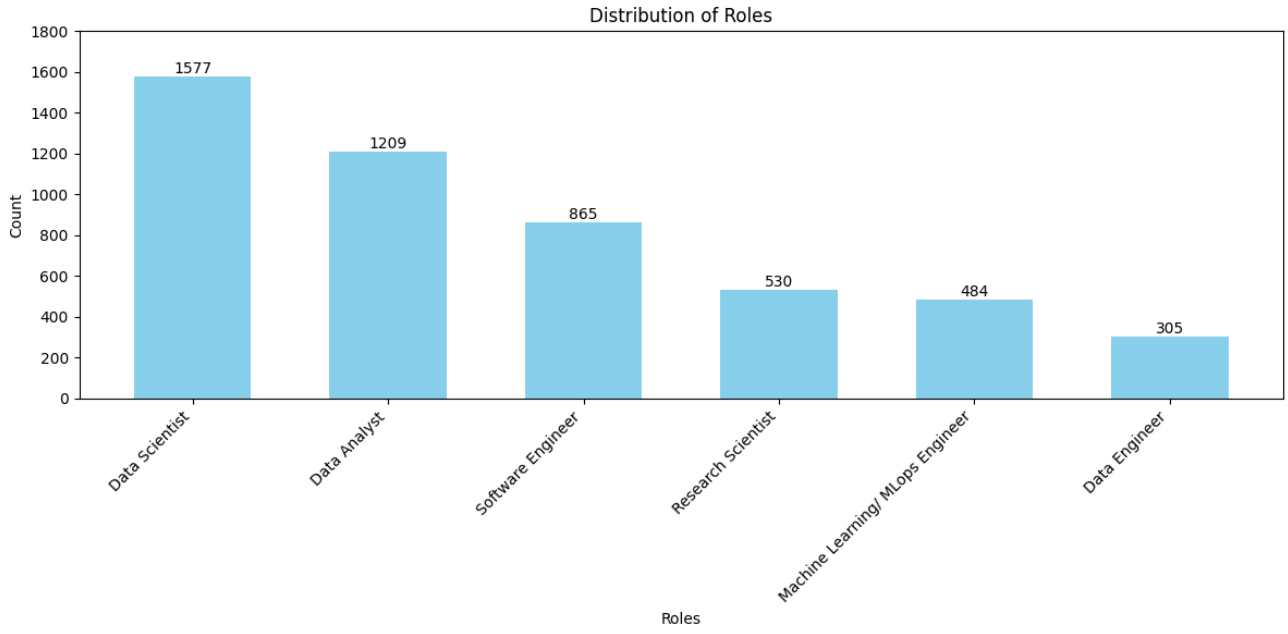


Figure 20: *Distribution of Roles After Data Cleaning*

Since the scope of the analysis is skill overlap in the technological related roles, only roles specifically related to this field were included. Roles like Statistician, Data Architect, Data Administrator, and Developer Advocate were excluded due to having fewer than 150

observations. The final roles selected for the analysis were: Data Scientist, Data Analyst, Software Engineer, Research Scientist, ML Engineer, and Data Engineer.

Dummy Variables: We converted categorical columns into dummy variables to facilitate modelling. These columns originally contained text responses that represented categories. We transformed each unique category into a separate binary (0/1) column. This process helps ML algorithms and statistical models interpret the data more effectively, since they typically cannot directly handle non-numeric or textual data.

A.1.2. Selected Questions for Analysis

Feature selection by correlation analysis: Features were selected based on their relevance as proxies for practical skills, focusing on activities and tools used regularly by respondents; a subsequent correlation analysis confirmed that no additional features with strong correlations were overlooked. Correlation was calculated following the Pearson's correlation coefficient (Sedgwick 2012):

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

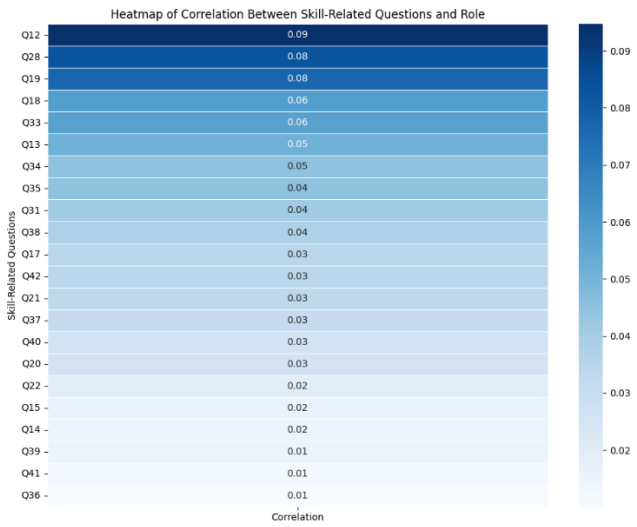


Figure 21: Correlation Between Skills-Related Question and Roles: *The heatmap shows the correlation between skill-related questions and roles, with darker shades indicating higher correlations.*

Appendix

Skills-related Questions used in the analysis

Q12: *What programming languages do you use on a regular basis?*

- Alternatives: Python, R, SQL, C, C#, Java, JavaScript, Bash, PHP, MATLAB, Julia, Go, None, Other.

Q13: *Which of the following integrated development environments (IDEs) do you use on a regular basis? (Select all that apply)*

- JupyterLab, RStudio, Visual Studio, Visual Studio Code (VSCode), PyCharm, Spyder, Notepad++, Sublime Text, Vim/Emacs, MATLAB, Jupyter Notebook, IntelliJ, None, Other.

Q15: *Do you use any of the following data visualization libraries on a regular basis? (Select all that apply)*

- Matplotlib, Seaborn, Plotly/Plotly Express, Ggplot/ggplot2, Shiny, D3.js, Altair, Bokeh, Geoplotlib, Leaflet/Folium, Pygal, Dygraphs, Highcharter, None, Other.

Q17: *Which of the following ML frameworks do you use on a regular basis? (Select all that apply)*

- Scikit-learn, TensorFlow, Keras, PyTorch, Fast.ai, Xgboost, LightGBM, CatBoost, Caret, Tidymodels, JAX, PyTorch Lightning, Huggingface, None, Other.

Q18: *Which of the following ML algorithms do you use on a regular basis? (Select all that apply)*

- Linear or Logistic Regression, Decision Trees or Random Forests, Gradient Boosting Machines (Xgboost, LightGBM, etc.), Bayesian Approaches, Evolutionary Approaches, Dense Neural Networks (MLPs, etc.), Convolutional Neural Networks, Generative Adversarial Networks, Recurrent Neural Networks, Transformer Networks (BERT,

GPT-3, etc.), Autoencoder Networks (DAE, VAE, etc.), Graph Neural Networks, None, Other.

Q19: Which categories of computer vision methods do you use on a regular basis? (Select all that apply)

- General-purpose image/video tools (PIL, cv2, skimage, etc.), Image segmentation methods (U-Net, Mask R-CNN, etc.), Object detection methods (YOLOv6, RetinaNet, etc.), Image classification networks (VGG, Inception, ResNet, etc.), Generative Networks (GAN, VAE, etc.), Vision transformer networks (ViT, DeiT, BiT, etc.), None, Other.

Q20: Which of the following NLP methods do you use on a regular basis? (Select all that apply)

- Word embeddings/vectors (GLoVe, fastText, word2vec), Encoder-decoder models (seq2seq, vanilla transformers), Contextualized embeddings (ELMo, CoVe), Transformer language models (GPT-3, BERT, XLNet, etc.), None, Other.

Q37: Do you use any of the following managed ML products on a regular basis? (Select all that apply)

- Amazon SageMaker, Azure ML Studio, Google Cloud Vertex AI, DataRobot, Databricks, Dataiku, Alteryx, RapidMiner, C3.ai, Domino Data Lab, H2O AI Cloud, No/None, Other.

A.2.1 Cosine similarity on Values Constrained between 0 and 1

The following part considers how the constrain on positive values in the calculation of average usage of skills across roles may inflate the cosine similarity scores. The equations show why we can expect average of cosine similarity scores of around 0.75 when number of features increases.

Appendix

Consider two vectors $\mathbf{a}$ and $\mathbf{b}$, each of dimension n, with their components independently drawn from a uniform distribution on $[0,1]$.

The cosine similarity is defined as: $"cos"\ (\theta) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$.re, $\mathbf{a}\cdot\mathbf{b} = \sum_{i=1}^{n} a_i\, b_i$, and $\|\mathbf{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}$ (similarly for $\mathbf{b}$).

Expected Values if each component a_i, b_i is uniform on [0,1]:

- **Mean of a single component:** $\mathbb{E}[a_i\ ] = 0.5$.
- **Mean of the product:** Since a_i and b_i are independent and identically distributed (i.i.d.): $\mathbb{E}[a_i b_i] = \mathbb{E}[a_i]\mathbb{E}[b_i] = 0.5 \times 0.5 = 0.25$.
- **Mean of the square of a component:** $\mathbb{E}[a_i^2] = \int_0^1 x^2\, dx = \frac{1}{3}$.

Expected Dot Product: For the entire vectors: $\mathbb{E}[\mathbf{a}\cdot\mathbf{b}] = \sum_{i=1}^{n} \mathbb{E}\,[a_i b_i] = n \times 0.25 = \frac{n}{4}$.

Expected Norm: The squared norm of $\mathbf{a}$ is: $\|\mathbf{a}\|^2 = \sum_{i=1}^{n} a_i^2$. The expectation is: $\mathbb{E}[\|\mathbf{a}\|^2] = \sum_{i=1}^{n} \mathbb{E}\,[a_i^2] = n \times \frac{1}{3} = \frac{n}{3}$. By the Law of Large Numbers and concentration results, for large n, $\|\mathbf{a}\|$ will be close to $\sqrt{\frac{n}{3}}$ with high probability. Similarly, $\|\mathbf{b}\|$ will also be about $\sqrt{\frac{n}{3}}$.

Typical cosine similarity: Putting these together: $"cos"\ (\theta) \approx \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \approx \frac{\frac{n}{4}}{\left(\sqrt{\frac{n}{3}}\right)\left(\sqrt{\frac{n}{3}}\right)} = \frac{\frac{n}{4}}{\frac{n}{3}} = \frac{3}{4} =$ 0.75. his rough calculation suggests that two random vectors in high dimensions, where each component is uniform on [0,1], are *not* orthogonal on average. In fact, their cosine similarity hovers around 0.75.

Appendix

Euclidean distance

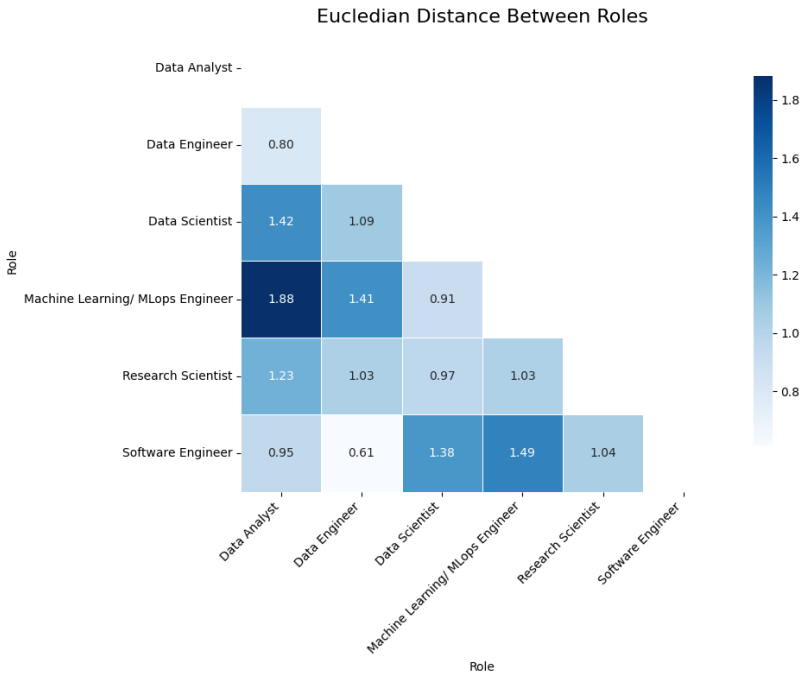


Figure 22: Euclidian distance Between Roles: *The heatmap shows the Euclidean distances between different roles, where shorter distances indicate higher similarity.*

Cosine similarity

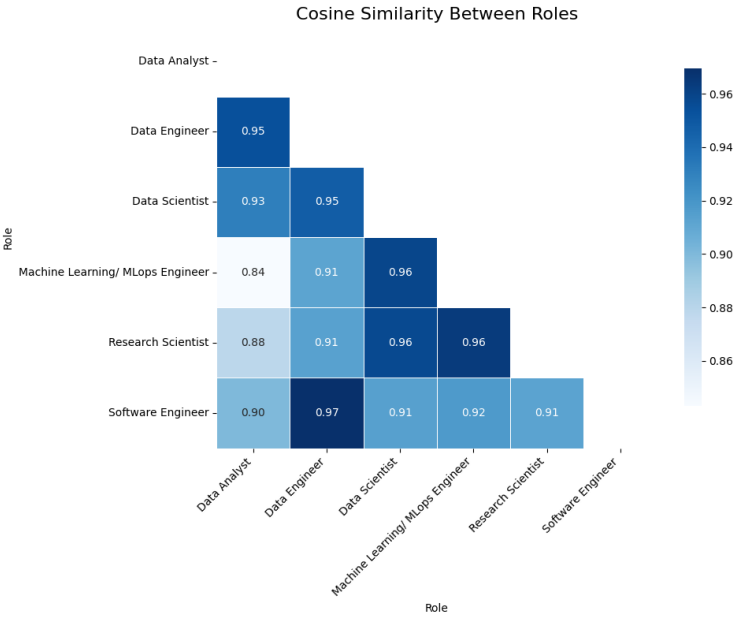


Figure 23: Cosine similarity Between Roles: *The heatmap shows the cosine similarity between different roles, where higher values indicate greater similarity.*

A.2.2 Skills Distribution Plots Between Roles

The following plots illustrate the differences in mean skill usage across various role pairs and skill categories, specifically for skills used by at least 50% of participants in either role. Differences that are statistically significant are highlighted with a red *. To determine statistical significance, a Chi-squared test was applied (Plackett 1982). The formula is given by:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

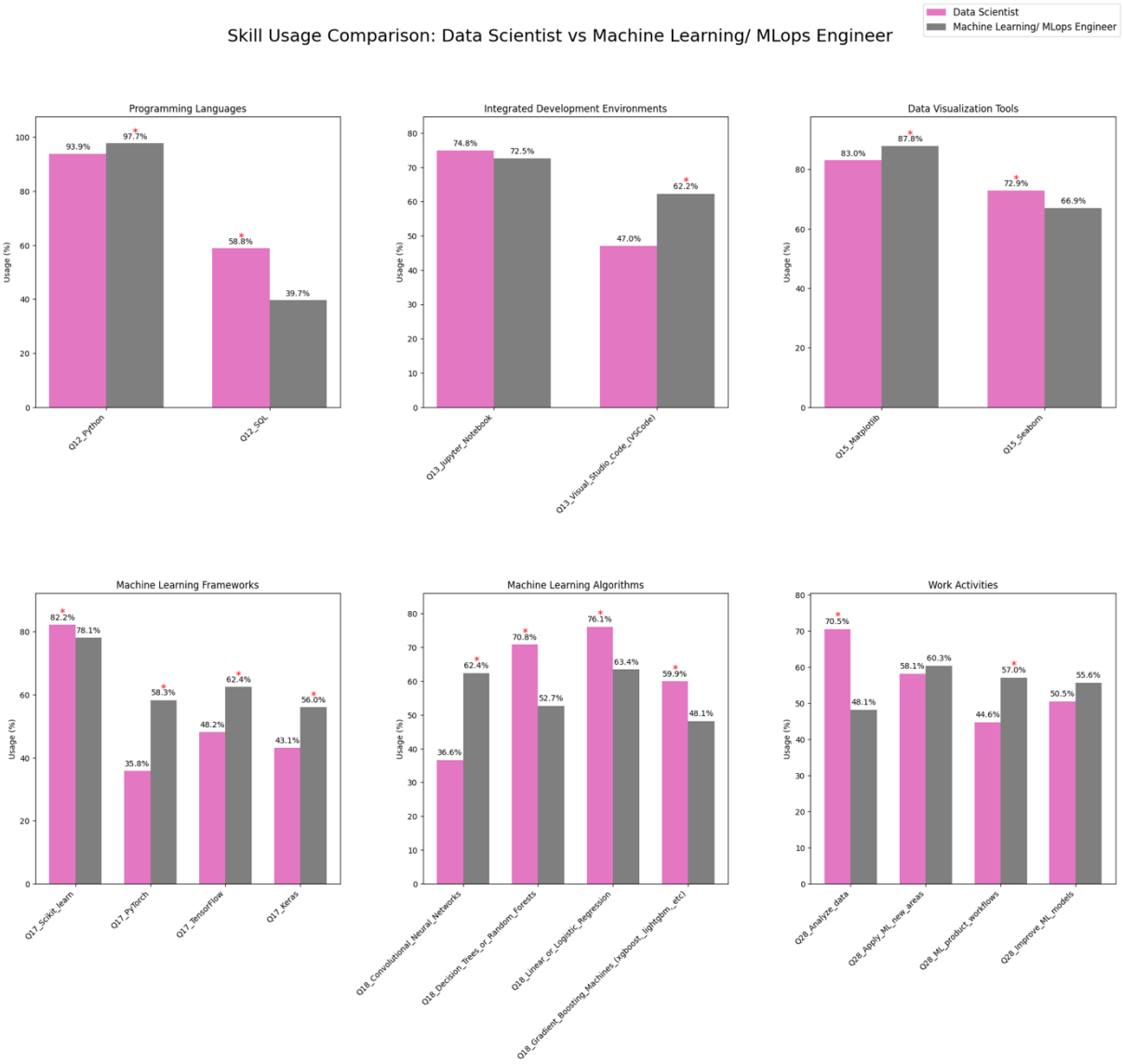


Figure 24: *Skill Usage Comparison Between Data Scientist and ML Engineer*

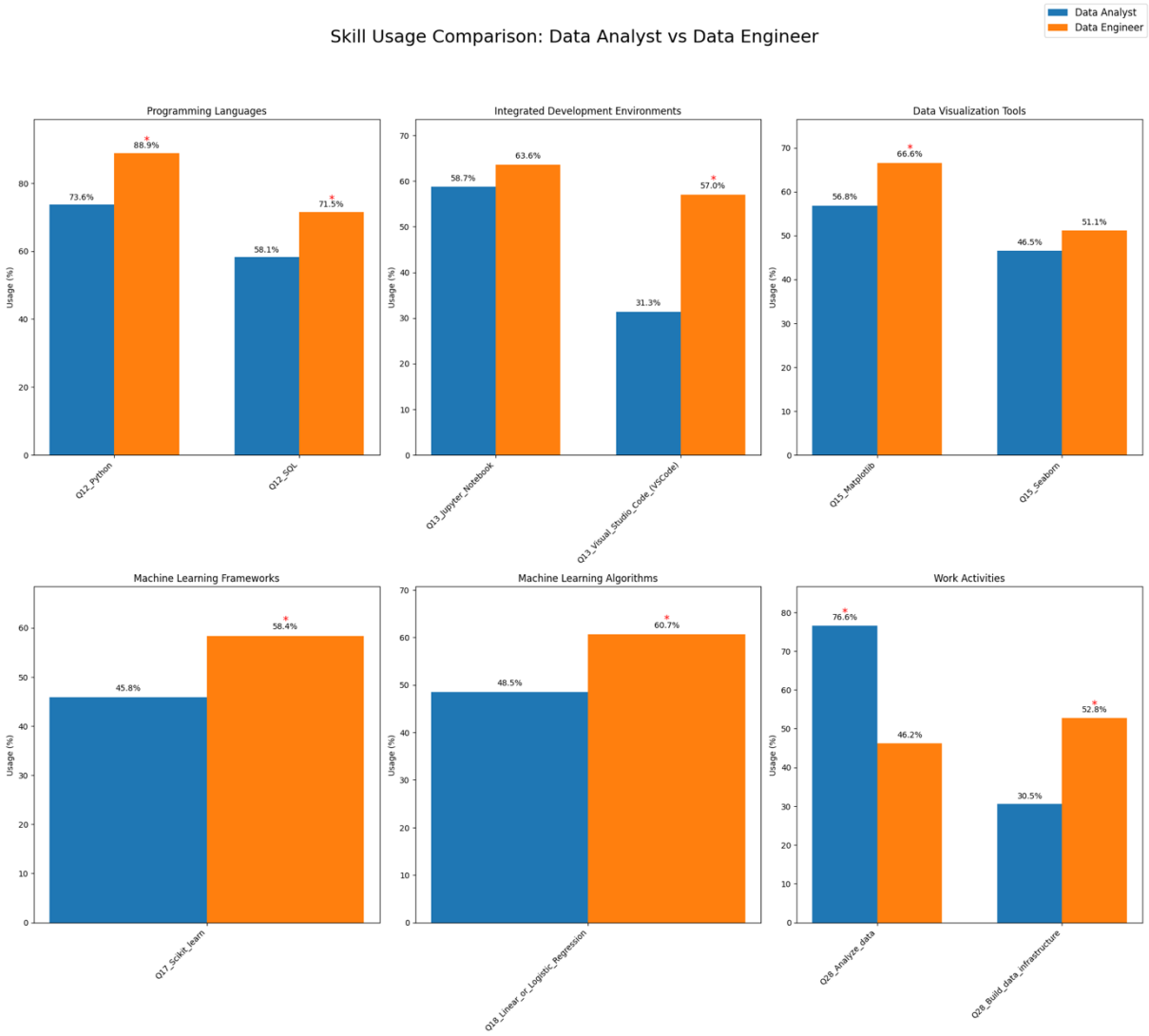


Figure 25: *Skill Usage Comparison Between Data Analyst and Data Engineer*

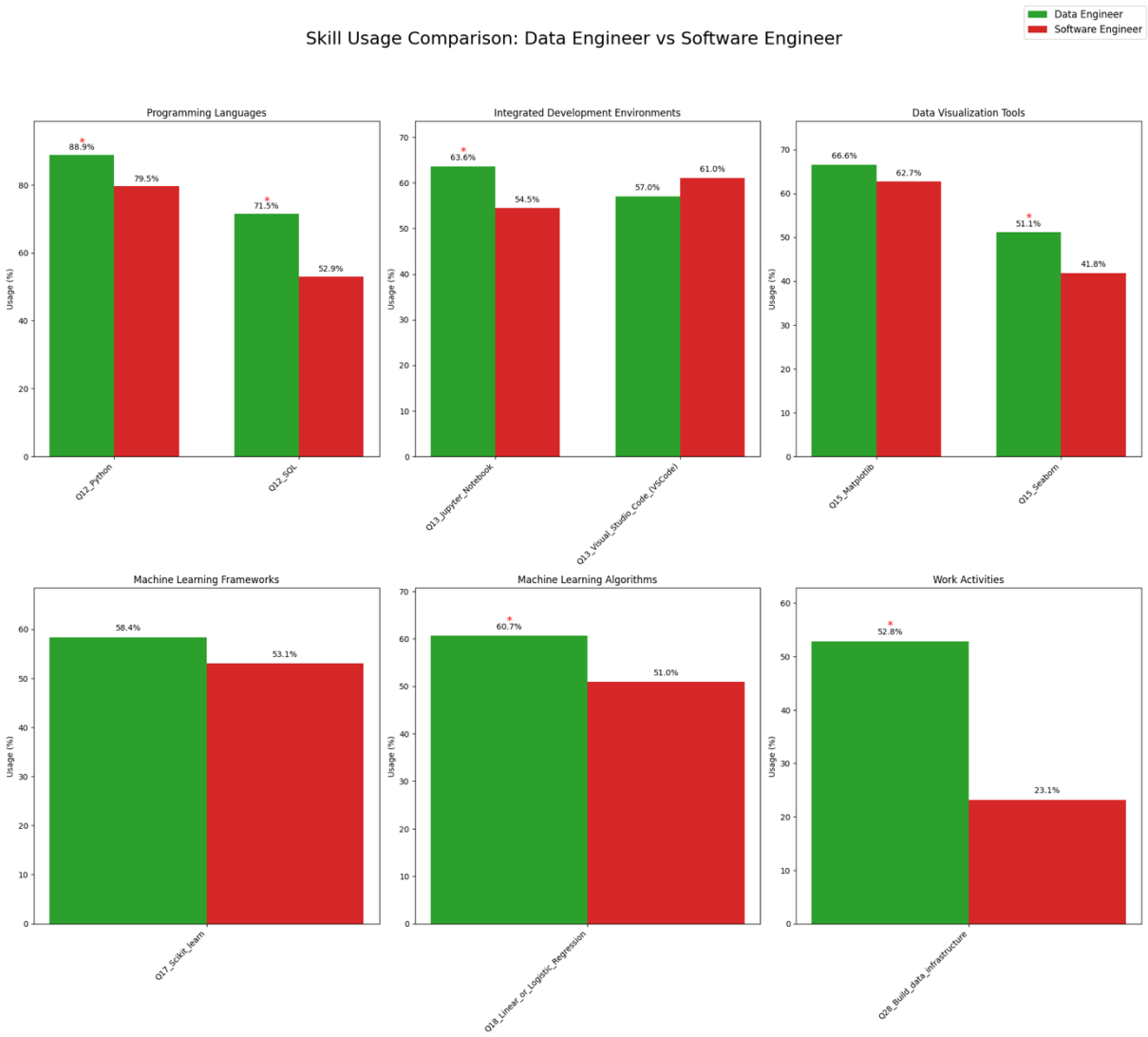


Figure 26: *Skill Usage Comparison Between Data Engineer and Software Engineer*

Appendix

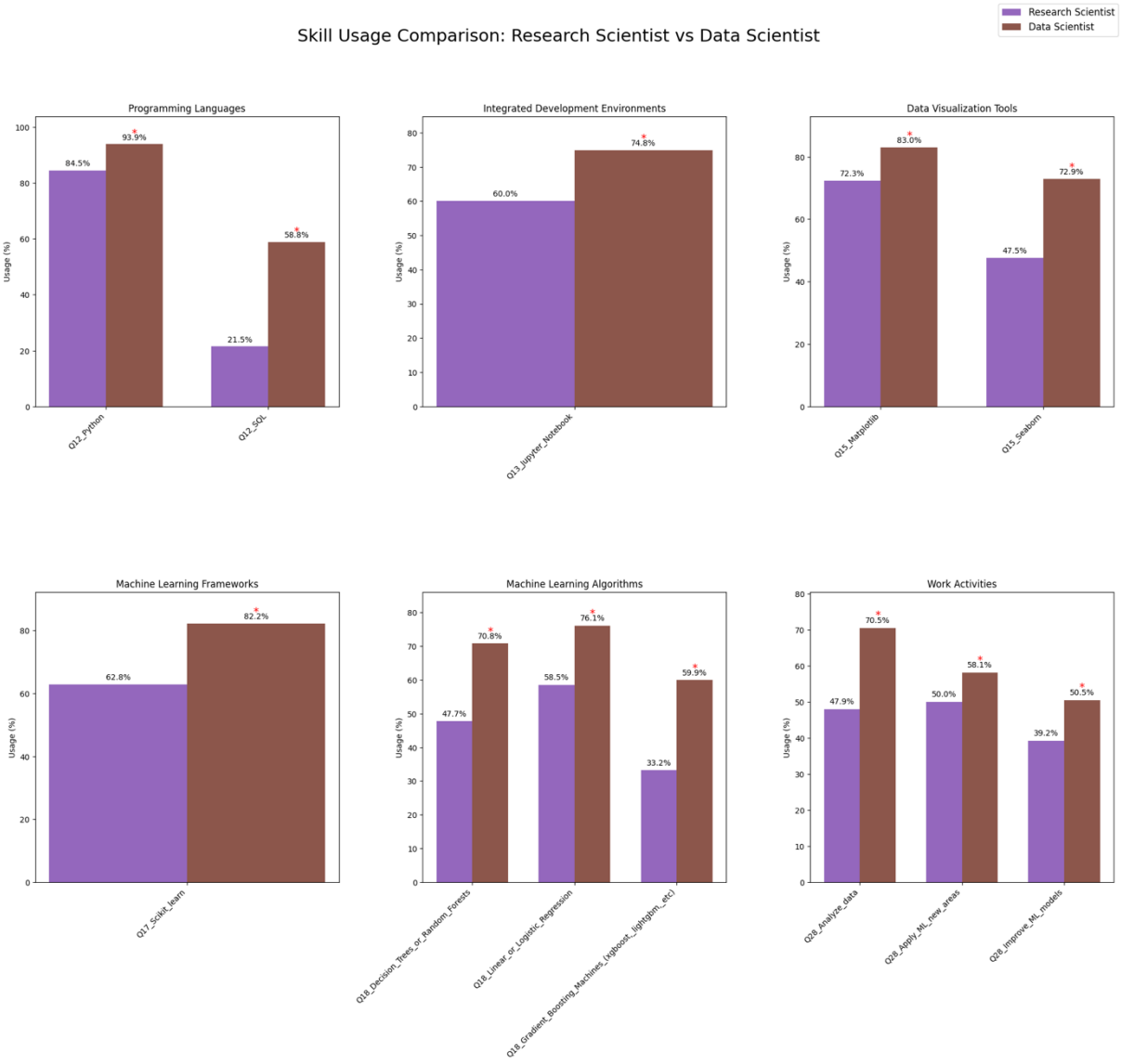


Figure 27: *Skill Usage Comparison Between Research Scientist and Data Scientist*

A.3 Industry Analysis

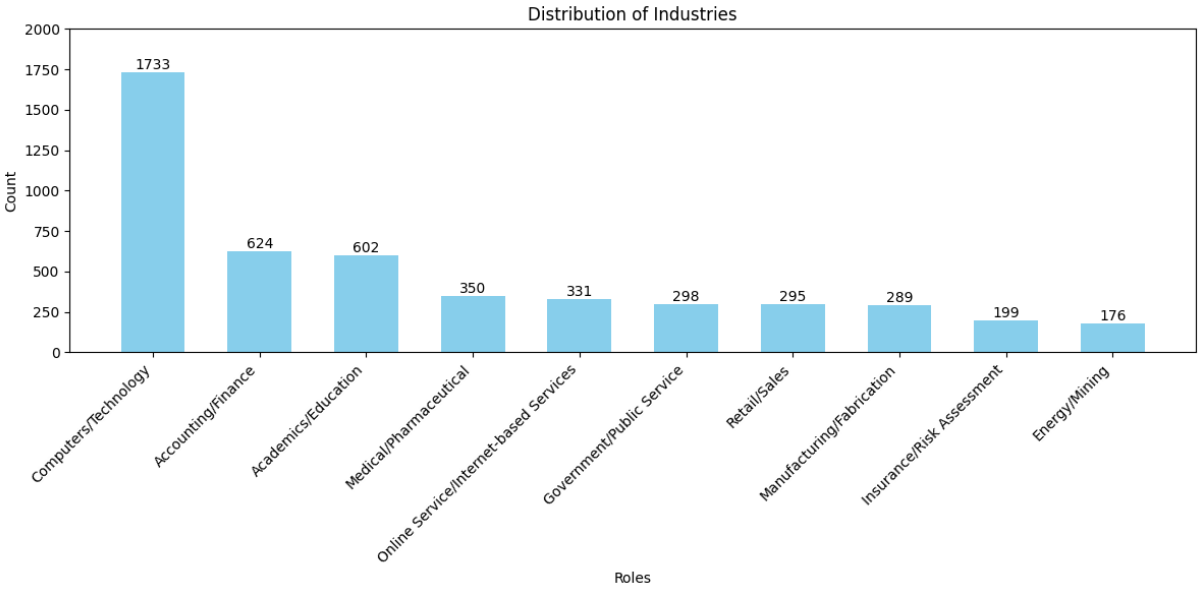


Figure 28: *Distribution of Industries After Cleaning*

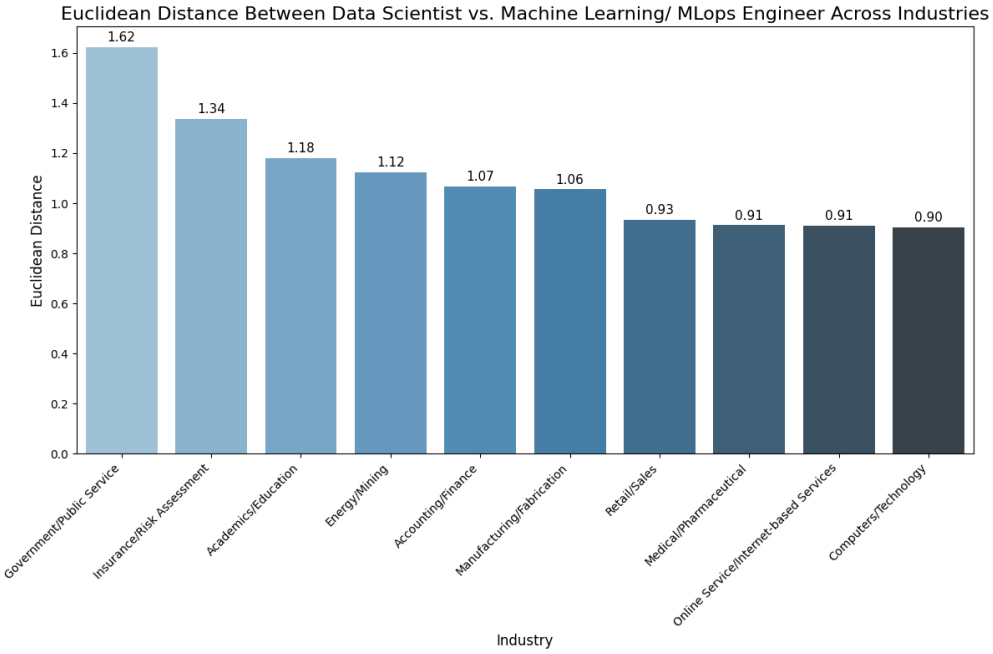


Figure 29: *Euclidean distance Across Industries for Data Scientist and ML Engineer*

Appendix

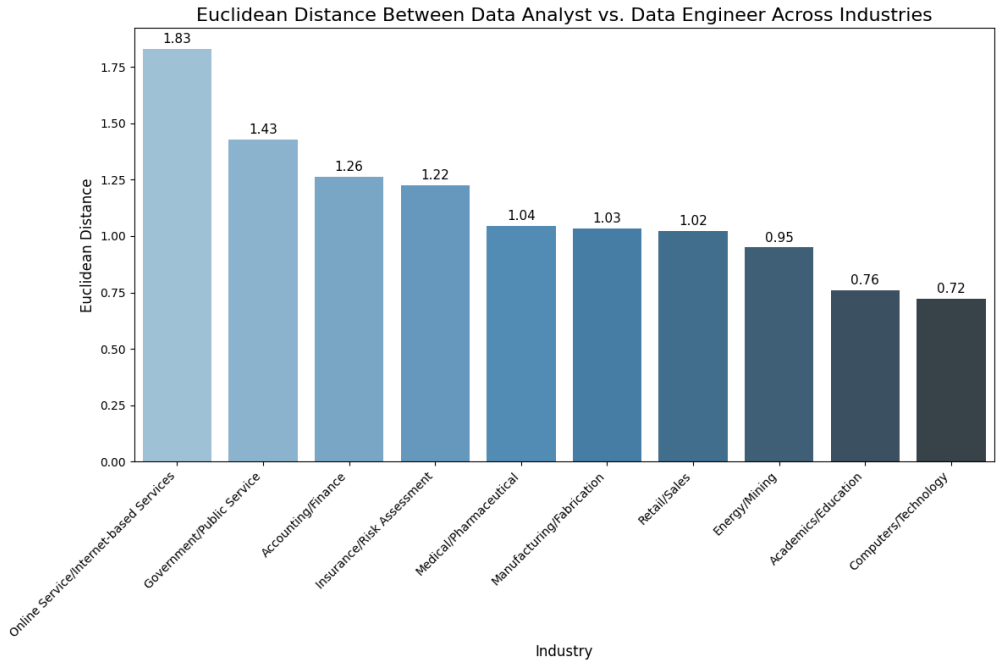


Figure 30: *Euclidean distance Across Industries for Data Analyst and Data Engineer*

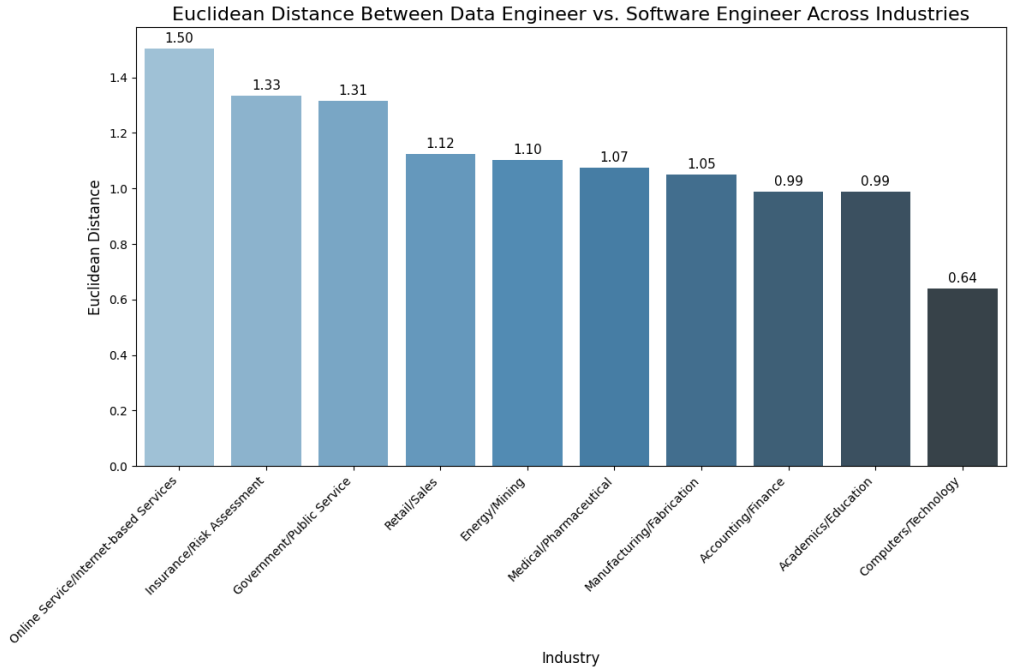


Figure 31: *Euclidean distance Across Industries for Data Engineer and Software Engineer*

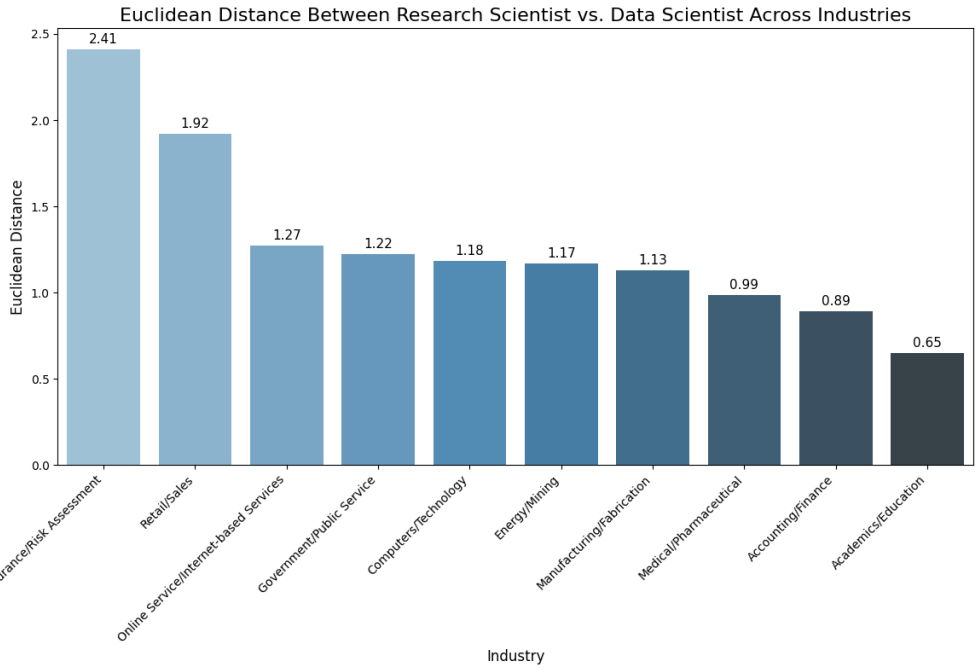


Figure 32: *Euclidean distance Across Industries for Research Scientist and Data Scientist*

A.4.2 O*NET data

Alternative Data Source for ML Engineer Skill Demand Analysis:

Since the ML Engineer role was not available in the O*NET database, we sourced relevant skill demand data from an external source. Specifically, we used the 2024 Most In-Demand ML Engineer Skills report by 365datascience. This report was selected because it is recent, aligns with the technical skills typically covered in ONET, and employs a comparable methodology, with findings derived from analysing 1,000 job postings (Magnet 2024).

Euclidean distances Skill Demand Analysis:

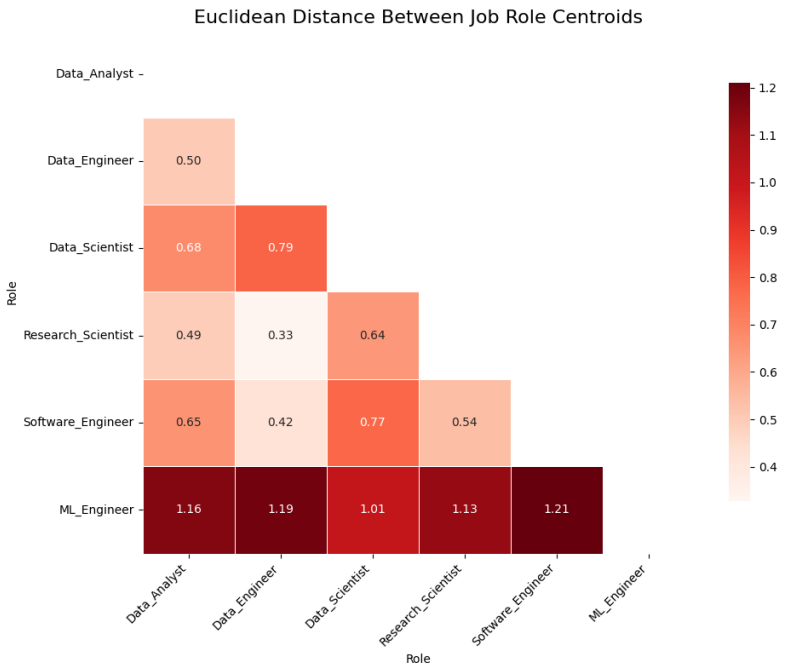


Figure 33: Euclidean distance Using O*NET Data: *The heatmap shows the Euclidean distances between different roles, where shorter distances indicate higher similarity.*

Appendix B – Model Development

B.1 Codebase overview

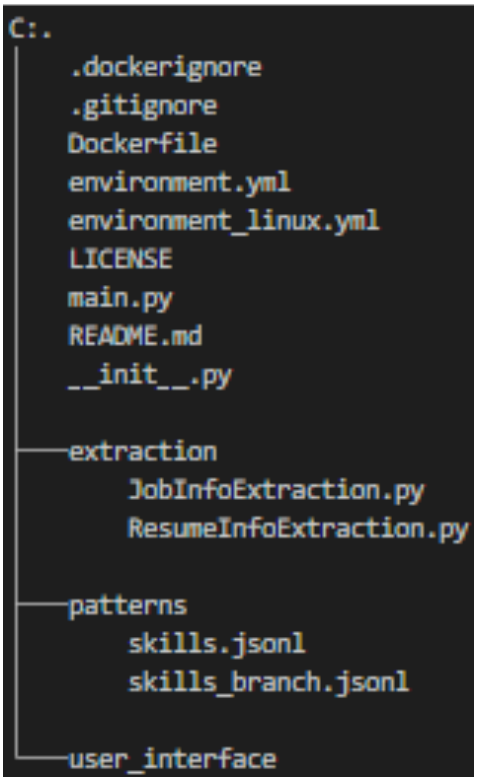


Figure 34: The following file tree shows the outline of the code files for the model. Main.py contains the core logic of the model. The extraction directory contains the modules for extracting info from résumés and job descriptions. The patterns directory contains the pattern files used by spaCy for the skill extraction – skills.jsonl is the pattern used for the model outlined in this section.

B.2 Résumé Parsing

Code 1: *This code processes resumes by extracting text from PDF files within a specified directory using the PyMuPDF library. Extracted data is stored in a structured pandas DataFrame, with columns for the candidate's name and raw resume text. Error handling ensures robustness in parsing.*

```
import os
import pandas as pd
import numpy as np
from extraction.ResumeInfoExtraction import ResumeInfoExtraction
from extraction.JobInfoExtraction import JobInfoExtraction
from source.schemas.resumeextracted import ResumeExtractedModel # Let's reintroduce later
from source.schemas.jobextracted import JobExtractedModel # Let's reintroduce later
import fitz  # PyMuPDF
from sentence_transformers import SentenceTransformer, CrossEncoder
from sklearn.metrics.pairwise import cosine_similarity
from openai import OpenAI
from tenacity import retry, wait_random_exponential, stop_after_attempt
import markdown
import warnings
import logging
import torch
import torch.nn.functional as F

# Handle annoying warnings
logging.getLogger('pypdf').setLevel(logging.ERROR)
warnings.filterwarnings("ignore")

# Get the absolute path of the root directory
ROOT_DIR = os.path.abspath(os.path.join(os.path.dirname(__file__), '..'))

# Paths to your pattern files
skills_patterns_path = os.path.join(ROOT_DIR, 'workproject_matching_algo','patterns', 'skills.jsonl')
skills_patterns_path_2 = os.path.join(ROOT_DIR, 'workproject_matching_algo','patterns',
'skills_branch.jsonl')

def get_resumes(directory):
    """ Function to parse and extract text from PDFs in a directory """

    def extract_pdf(path):
        """ Helper function to extract the text from the PDFs using the PyMuPDF library"""
        try:
            with fitz.open(path) as doc:
                text = ''.join(page.get_text() for page in doc)
            return text
        except Exception as e:
            logging.error(f"Error processing {path}: {e}")
            return ""

    # initialize empty dictionary
    dic = {}

    # Iterate over all files in the directory
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)

        # Extract text from pdf if file is pdf
        if os.path.isfile(file_path) and filename.endswith(".pdf"):
            name = filename.rstrip(".pdf")
            resume_text = extract_pdf(file_path)
            dic[name] = [resume_text]

    # Create a pandas dataframe from the parsed PDFs
    df = pd.DataFrame(dic).T
    df.reset_index(inplace=True)
    df.rename(columns={"index": "name", 0: "raw"}, inplace=True)

    return df
```

B.3.1 Skill Extraction (from main.py)

Code 2: *These functions extract key skills from resumes and job descriptions. The resume_extraction function leverages the ResumeInfoExtraction class and predefined skill patterns to process candidate resumes. Similarly, job_info_extraction uses the JobInfoExtraction class to extract relevant skills from job descriptions. Both functions return structured DataFrames for downstream analysis.*

```
def resume_extraction(resumes):
    """ function to extract the relevant skills from a resume """
    names = resumes[["name"]]
    resume_extraction = ResumeInfoExtraction(skills_patterns_path_2, names)
    resumes_df = resume_extraction.extract_entities(resumes)
    return resumes_df


def job_info_extraction(jobs):
    """ function to extract the relevant skills from a job description """
    job_extraction = JobInfoExtraction(skills_patterns_path)
    job_df = job_extraction.extract_entities(jobs)
    return job_df
```

B.3.2 Skill Extraction (from *extraction* directory)

Code 3: *The JobInfoExtraction class uses spaCy's EntityRuler to identify skills in job descriptions based on predefined patterns. It includes methods to match skills (match_skills_by_spacy) and process a DataFrame of job descriptions (extract_entities). The resulting output adds a Skills column to the input DataFrame, listing extracted skills for each job description.*

```
from spacy.lang.en import English

class JobInfoExtraction:
    """
    A class to extract job-related skills information using spaCy's EntityRuler.

    Attributes:
        skills_patterns_path (str): Path to the JSON file containing skill patterns.
        nlp (Language): spaCy language model with the EntityRuler pipeline.
    """

    def __init__(self, skills_patterns_path):
        """
        Initializes the JobInfoExtraction class with the specified skills patterns.

        Args:
            skills_patterns_path (str): Path to the JSON file with skill patterns.
        """
        self.skills_patterns_path = skills_patterns_path
        self.nlp = English()
        # Add EntityRuler to the spaCy pipeline and load patterns from the specified path
        ruler = self.nlp.add_pipe("entity_ruler")
        ruler.from_disk(self.skills_patterns_path)

    def match_skills_by_spacy(self, job, display=False):
        """
        Extracts skills mentioned in a job description using spaCy.

        Args:
            job (str): Job description text.
            display (bool): Whether to print the extracted skills (default: False).

        Returns:
            list: A list of unique skill labels extracted from the job description.
        """
        doc = self.nlp(job)  # Process the job description
        job_skills = []

        for ent in doc.ents:
            labels_parts = ent.label_.split('|')
```

```
            # Check if the label indicates a skill
            if labels_parts[0] == 'SKILL':
                skill_name = labels_parts[1].replace('-', ' ')
                if display:
                    print((ent.text, ent.label_))
                # Add the skill to the list if not already present
                if skill_name not in job_skills:
                    job_skills.append(skill_name)

        return job_skills

    def extract_entities(self, jobs):
        """
        Processes a DataFrame of job descriptions and extracts skills.

        Args:
            jobs (DataFrame): A pandas DataFrame with a column named 'raw' containing job descriptions.

        Returns:
            DataFrame: The input DataFrame with an additional 'Skills' column listing extracted skills.
        """
        # Clean the job descriptions and extract skills
        jobs['Skills'] = (
            jobs['raw']
            .str.replace('.', ' ')
            .str.replace('\n', ' ')
            .apply(self.match_skills_by_spacy)
        )
        return jobs
```

Code 4: *The ResumeInfoExtraction class uses spaCy's EntityRuler to extract skills from resumes based on predefined patterns. It includes methods to match skills (match_skills_by_spacy) and process a DataFrame of resumes (extract_entities). The output adds a Skills column to the input DataFrame, listing unique skills identified in each resume.*

```
from spacy.lang.en import English

class ResumeInfoExtraction:
    """
    A class to extract skills information from resumes using spaCy's EntityRuler.

    Attributes:
        skills_patterns_path (str): Path to the JSON file containing skill patterns.
        name (str): Name or identifier for this instance of extraction.
        nlp (Language): spaCy language model with the EntityRuler pipeline.
    """

    def __init__(self, skills_patterns_path, name):
        """
        Initializes the ResumeInfoExtraction class with the specified skills patterns and name.

        Args:
            skills_patterns_path (str): Path to the JSON file with skill patterns.
            name (str): Name or identifier for this extraction process.
        """
        self.name = name
        self.skills_patterns_path = skills_patterns_path
        self.nlp = English()
        # Add EntityRuler to the spaCy pipeline and load patterns from the specified path
        ruler = self.nlp.add_pipe("entity_ruler")
        ruler.from_disk(self.skills_patterns_path)

    def match_skills_by_spacy(self, resume, display=False):
        """
        Extracts skills mentioned in a resume using spaCy.

        Args:
            resume (str): Resume text.
            display (bool): Whether to print the extracted skills (default: False).

        Returns:
            list: A list of unique skill labels extracted from the resume.
```

```
        """
        doc = self.nlp(resume)  # Process the resume text
        resume_skills = []

        for ent in doc.ents:
            labels_parts = ent.label_.split('|')
            # Check if the label indicates a skill
            if labels_parts[0] == 'SKILL':
                skill_name = labels_parts[1].replace('-', ' ')
                if display:
                    print((ent.text, ent.label_))
                # Add the skill to the list if not already present
                if skill_name not in resume_skills:
                    resume_skills.append(skill_name)

        return resume_skills

    def extract_entities(self, resumes):
        """
        Processes a DataFrame of resumes and extracts skills.

        Args:
            resumes (DataFrame): A pandas DataFrame with a column named 'raw' containing resume texts.

        Returns:
            DataFrame: The input DataFrame with an additional 'Skills' column listing extracted skills.
        """
        # Clean the resume texts and extract skills
        resumes['Skills'] = (
            resumes['raw']
            .str.replace('. ', ' ')  # Replace periods with spaces
            .str.replace('\n', ' ')  # Replace newlines with spaces
            .apply(self.match_skills_by_spacy)
        )
        return resumes
```

B.4 Bag-of-skills with Special token

Code 5: *Comparison of similarity scores and ranking with and without a special token between skills during embedding. Adding a special token improves the model's ability to distinguish individual skills, witnessed by the model's ability to correctly rank the synthetic resumes in descending order.*

```
    print("bag of skills - with special token:")
    res, *rest = calc_similarity(df_resumes, df_jobs); print(res.sort_values("rank"))
9]  ✓ 1.8s

bag of skills - with special token:
                                    name job_id  similarity_score  rank  \
4           4_Skills_AWS_Git_Python_Azure      0          0.507241   1.0
3  3_Skills_UNRELATED_AWS_Power BI_Tableau      0          0.453915   2.0
2                3_Skills_AWS_Git_Python      0          0.428471   3.0
1                       2_Skills_AWS_Git      0          0.427074   4.0
0                           1_Skill_AWS      0          0.343247   5.0

  interview_status
4         Selected
3         Selected
2         Selected
1     Not Selected
0     Not Selected

    print("bag of skills:")
    res, *rest = calc_similarity(df_resumes, df_jobs); print(res.sort_values("rank"))
3]  ✓ 2.0s

bag of skills:
                                    name job_id  similarity_score  rank  \
4           4_Skills_AWS_Git_Python_Azure      0          0.365599   1.0
3  3_Skills_UNRELATED_AWS_Power BI_Tableau      0          0.351982   2.0
0                           1_Skill_AWS      0          0.331589   3.0
2                3_Skills_AWS_Git_Python      0          0.316115   4.0
1                       2_Skills_AWS_Git      0          0.293126   5.0

  interview_status
4         Selected
3         Selected
0         Selected
2     Not Selected
1     Not Selected
```

B.5 Embedding and Similarity Analysis

Code 6: *The calc_similarity_baseline function computes cosine similarity between precomputed MPNET embeddings of skills for job descriptions and applicants. Skills are embedded as a single sequence with a special token inserted between each skill to enhance recognition and ranking accuracy. This approach reduces computational inefficiencies while improving candidate selection precision.*

```
def calc_similarity_baseline(applicant_df, job_df, N=3, parallel=False):
    """Calculate cosine similarity based on MPNET embeddings of combined skills."""

    # Initialize the model
    model = SentenceTransformer('all-mpnet-base-v2')
    model.max_seq_length = 75
    model.tokenizer.padding_side="right"
    model.eval()

    def add_eos(input_examples):
        """ helper function to add special tokens between each skills"""
        input_examples = [input_example + model.tokenizer.eos_token for input_example in
input_examples]
        return input_examples

    # Precompute job embeddings
    job_df['Skills_Text'] = job_df['Skills'].apply(add_eos) # add special tokens
    job_df['Skills_Text'] = job_df['Skills_Text'].apply(lambda x: ' '.join(sorted(set(x))) if
isinstance(x, list) else '') # concatenate into one string
    job_embeddings = model.encode(job_df['Skills_Text'].tolist()) # encode job descriptions
    # Precompute applicant embeddings
    applicant_df['Skills_Text'] = applicant_df['Skills'].apply(add_eos) # add special tokens
    applicant_df['Skills_Text'] = applicant_df['Skills_Text'].apply(lambda x: ' '.join(sorted(set(x)))
if isinstance(x, list) else '') # concatenate into one string
    applicant_embeddings = model.encode(
        applicant_df['Skills_Text'].tolist(),
        batch_size=32,
        num_workers=os.cpu_count() // 2 if parallel else 0, # parallelize
        show_progress_bar=False
    )

    # Compute cosine similarity matrix
    similarity_matrix = cosine_similarity(job_embeddings, applicant_embeddings)

    # Create a DataFrame from the similarity matrix
    similarity_df = pd.DataFrame(similarity_matrix.T, index=applicant_df['name'], columns=job_df.index)
    similarity_df = similarity_df.reset_index().melt(id_vars='name', var_name='job_id',
value_name='similarity_score')
    similarity_df['rank'] = similarity_df.groupby('job_id')['similarity_score'].rank(ascending=False)
    similarity_df['interview_status'] = similarity_df['rank'].apply(lambda x: 'Selected' if x <= N else
'Not Selected')

    return similarity_df.sort_values("rank")
```

Appendix C - Synthetic Data Generation

C.1 EDA and Data Preprocessing of Job Descriptions Dataset

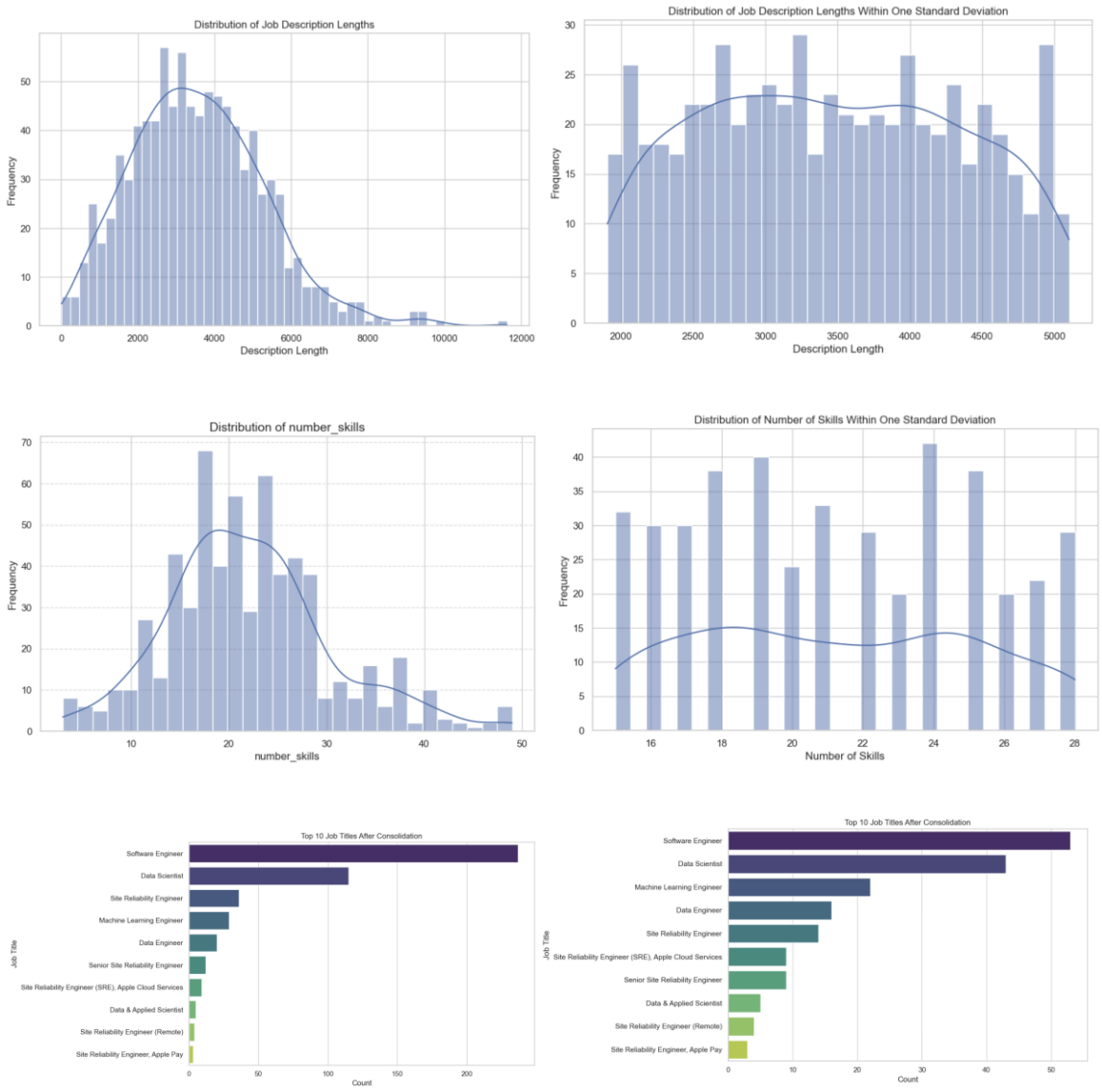


Figure 34: *Comparison of Job Description Length, Number of Skills, and Distribution of Roles before (left) and after (right) Data Preprocessing*

C.2 Format of Final Job Descriptions Dataset

	Job_ID	Job_title	Job_description
0	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...
1	2	Front End Cloud Software Developer	Candidate Requirements Must be physically loca...
2	3	Site Reliability Engineer	In Microsoft Health Next, our team is transfor...

Table 13: *Final Dataset holding 100 Job Descriptions (first three rows displayed)*

C.3 System and User Messaged used for Prompts

The following system and user messages were used for prompting the generation of good and bad fit résumés of the final selected dataset *df_synth_data_2_100* used for model testing and evaluation:

Prompt for Good Fit Résumés:

System Message:

"You are the world's best AI HR EXPERT that generates realistic resumes of different individuals which could be real persons, for a given job description from the tech sector.

The goal is to create clearly differentiable resumes all having a ***very good fit*** *with the provided job description only based on skill alignment.*

*The fit **must not depend** on any other factor except of the skills mentioned in the resume and the job description.*

Do not include any note or explanation of how you generate the resumes."

User Message:

"Generate a resume for an unique candidate applying for the following tech job. Make sure the skill alignment between the resume and the job description is ***very good*** *in terms of exact skill matches and related skills."*

Prompt for Bad Fit Résumés:

System Message:

"You are the world's best AI HR EXPERT that generates realistic resumes of different individuals which could be real persons, for a given job description from the tech sector.

The goal is to create clearly differentiable resumes all having a ***moderate fit*** *with the provided job description only based on skill alignment.*

*The fit **must not depend** on any other factor except of the skills mentioned in the resume and the job description.*

Do not include any note or explanation of how you generate the resumes."

User Message:

"Generate a resume for an unique candidate applying for the following tech job. Make sure the skill alignment between the resume and the job description is ***moderate*** *in terms of exact skill matches and related skills."*

C.4 Synthetic Dataset df_synth_data

	JD_ID	JD_title	JD_text	CV_ID	CV_text	label
0	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	11	**John Doe** \n123 Tech Lane \nPleasanton, C...	1
1	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	12	**John Smith** \n[LinkedIn Profile] | [Github...	1
2	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	13	**John D. Anderson** \n123 Tech Lane \nSan F...	1
3	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	14	**John Anderson** \n123 Tech Avenue \nPleasa...	0
4	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	15	**John Smith** \n123 Tech Lane \nSan Francis...	0
5	1	Senior Site Reliability Engineer	About the Team Workday is building a new SRE t...	16	**Jane Doe**\n\n[Your Address] \n[City, State...	0

Table 14: Synthetic Dataset "df_synth_data": *The first dataset holding 100 job descriptions with 3 good and 3 bad fit résumés for each job description (first six Rows displayed). The format of the dataset stays consistent for the versions that were created after based on refined prompts.*

C.5 Similarity Score Distribution Plots

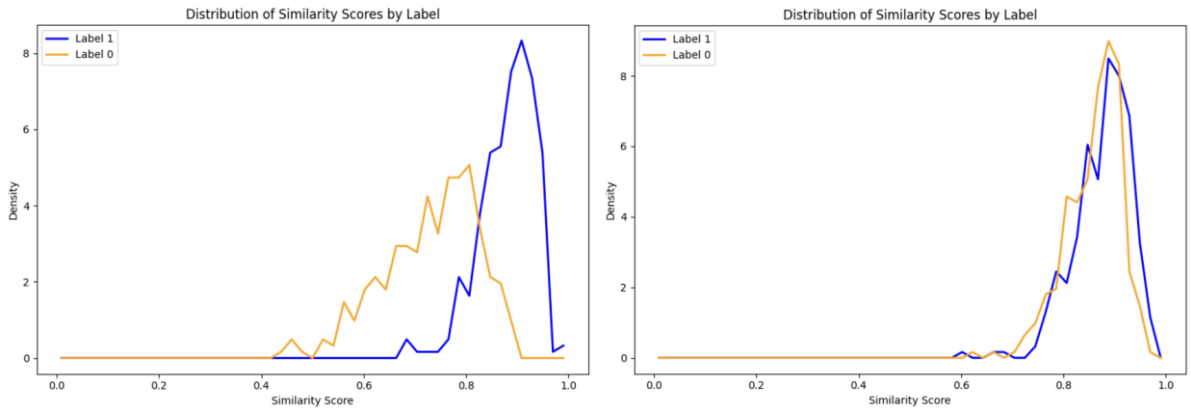


Figure 35: *Distribution of Similarity Scores using the initial df_synth_data (left) vs. the final dataset df_synth_data_2_100 (right)*

Appendix D - Evaluation

D.1 Detailed Explanation of Evaluation Metrics

To comprehensively evaluate the performance of the Skillmatch.ai pipeline, we relied on a combination of binary classification metrics and a custom-designed similarity metric. Each metric addresses specific aspects of model performance, ensuring a balanced assessment of predictive accuracy, operational efficiency, and candidate differentiation.

Precision (Powers 2020)

Definition: Precision measures the proportion of correctly identified Good Fit candidates (true positives) among all candidates predicted as Good Fit (true positives + false positives).

Formula:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Importance: Precision ensures that the model recommends suitable candidates, minimizing the risk of selecting unsuitable candidates (false positives). In recruitment systems, low precision could result in wasted interview resources, poor hiring decisions, and reduced organizational productivity.

Use in Evaluation: Precision is particularly critical when hiring processes prioritize candidate quality over quantity. A high precision score indicates the model's reliability in suggesting relevant résumés that match job requirements.

2. Recall (Powers 2020)

Definition: Recall assesses the proportion of actual **Good Fit candidates** (true positives) correctly identified by the model out of all actual Good Fit candidates (true positives + false negatives).

Formula:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Importance: Recall ensures that qualified candidates are not overlooked during the matching process. Missing strong candidates (false negatives) can lead to unfilled positions or suboptimal hiring outcomes, particularly in competitive job markets.

Use in Evaluation: Recall is vital for ensuring that the pipeline does not miss qualified talent, supporting comprehensive candidate coverage.

3. F1-Score (Powers 2020)

Definition: The F1-Score is the **harmonic mean** of precision and recall, combining both metrics into a single value to measure overall performance.

Formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Importance: F1-Score balances the trade-off between precision and recall, making it particularly valuable in recruitment systems where both metrics are critical. A high F1-Score reflects the model's ability to correctly identify **Good Fit candidates** while minimizing overlooked talent and irrelevant suggestions.

Use in Evaluation: The F1-Score is our primary evaluation metric. Calculated as the harmonic mean of precision and recall, it balances correctly identifying suitable candidates with minimizing overlooked talent. A balance between recall and precision is required to avoid overlooking talent while minimizing irrelevant recommendations. This makes it especially suitable for recruitment, where precision and recall are critical to reliable hiring decisions.

4. Mean Similarity Score Difference (self-designed)

Definition: The **Mean Similarity Score Difference** evaluates the model's ability to distinguish between **Good Fit (Label 1)** and **Bad Fit (Label 0)** candidates by calculating the average difference in similarity scores.

Formula:

$$\text{Mean SC Difference} = \frac{1}{n} \sum_{i=1}^{n} (\text{SC-Score for Good Fit}_i - \text{SC-Score for Bad Fit}_i)$$

Importance: This metric measures how well the model differentiates between candidates, with higher values indicating clearer distinctions. A higher **Mean SC Difference** improves the reliability of candidate ranking and ensures that the best-fit candidates are clearly prioritized.

Use in Evaluation: Models with a higher Mean SC Difference deliver sharper separation between suitable and unsuitable candidates, increasing recruiter confidence in the recommendations. It complements the F1-Score by providing additional insights into the quality of model outputs.

Rationale for Choosing the Mean Similarity Score Difference

The Mean Similarity Score Difference was introduced as a critical evaluation metric due to its ability to **quantify class separation** beyond binary classification outputs. While F1-Score

provides an aggregate measure of prediction accuracy, it does not capture how well the model distinguishes between candidates within the similarity space. Recruitment systems require not only accurate predictions but also clear prioritization of top candidates.

By calculating the average similarity difference between Good Fit and Bad Fit résumés, this metric ensures that top-ranked candidates are sufficiently separated from less suitable applicants, improving the overall quality and interpretability of model outputs.

D.2 Performance Analysis Results

This appendix provides visualizations used to analyse the performance of the Skillmatch.ai pipeline across all iterations and components. Each performance analysis includes the following four visualizations:

1. **Confusion Matrix**: Displays the classification performance by comparing predicted and actual labels, highlighting correct predictions and misclassifications for good-fit (Label 1) and bad-fit (Label 0) résumés.
2. **Distribution of Similarity Scores**: Shows the separation of similarity scores for good-fit (Label 1) and bad-fit (Label 0) résumés, providing a visual measure of how well the model distinguishes between candidates.
3. **Performance Metrics Table**: Summarizes key metrics, including the F1-Score, Mean Similarity Score Difference, Accuracy, and Processing Time, offering a quantitative evaluation of the model's performance.

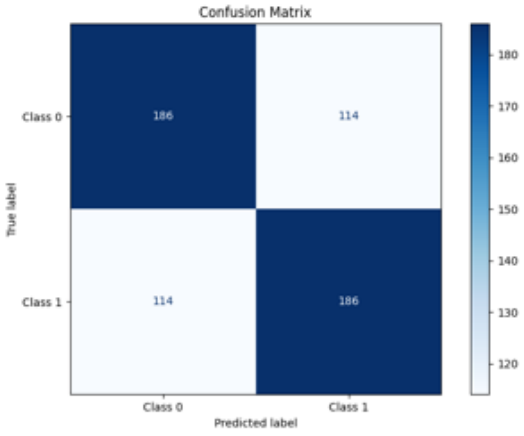
These visualizations are presented for each iteration and evaluated component, supporting the findings discussed in **Section 7.3**. The complete evaluation pipeline, including code and datasets, can be found in the evaluation folder of our GitHub repository:

https://github.com/victorbjorsvik/workproject_matching_algo.git.

D.2.1 Format of Skill Representation for Embeddings

This appendix provides a comprehensive overview of the performance evaluation of the **Bag-of-Skills (BOS)** and **Skill-by-Skill (SBS)** approaches, including supporting visualizations and detailed metrics.

Bag-of Skills: This figure presents the performance analysis of the Bag-of-Skills (BOS) approach. The confusion matrix shows significant misclassification between good-fit (Label 1) and bad-fit (Label 0) résumés, resulting in a lower F1-Score of 0.62. The precision-recall curve further illustrates the limited precision of the BOS approach. Additionally, the similarity score distribution (bottom right) reveals significant overlap between Labels 0 and 1, reducing the model's ability to differentiate between good and poor fits effectively.



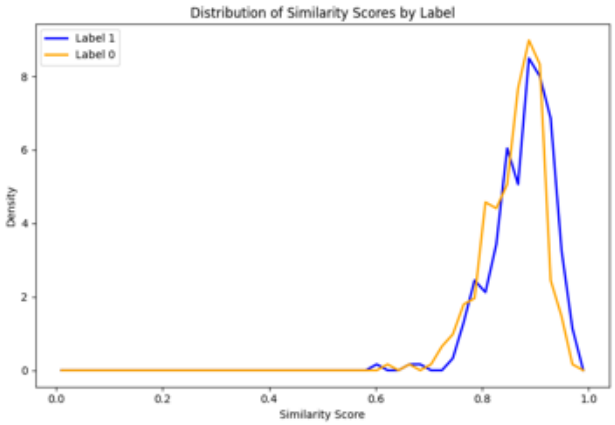


Figure 36: *Bag-of-Skills – Confusion Matrix*

Figure 37: *Bag-of-Skills - Distribution of Similarity Score by Labels*

Skill By Skill: This figure presents the performance analysis of the Skill-by-Skill (SBS) approach. The confusion matrix indicates improved classification performance, with fewer misclassifications compared to BOS. The SBS approach achieves a significantly higher F1-Score of 0.7471, as shown in the detailed metrics. The precision-recall curve highlights enhanced precision across recall values. The similarity score distribution (bottom right) shows

clearer separation between good-fit (Label 1) and bad-fit (Label 0) résumés, validating the improved performance of the SBS approach.

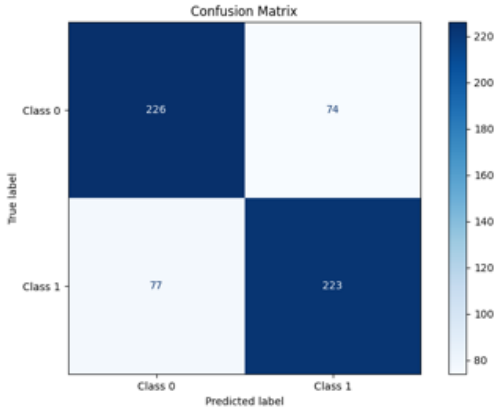


Figure 38: *Skill-by-Skills – Confusion Metrix*

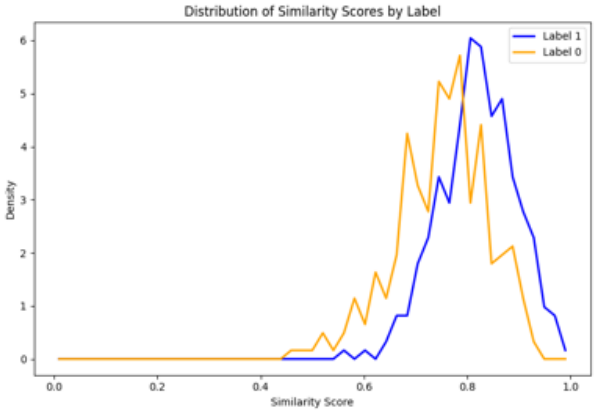


Figure 39: *Skill-by-Skills - Distribution of Similarity Score by Labels*

The evaluation of skill representation formats clearly demonstrates the advantages of the Skill-by-Skill approach over the Bag-of-Skills method. While BOS offers faster processing times, SBS achieves significantly better accuracy, precision, and class separation. These findings validate the adoption of the Skill-by-Skill format for subsequent iterations of the model pipeline, with further optimization opportunities through hyperparameter tuning.

Approach	*Processing Time*	*Mean SC-Diff.*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Bag-of-Skill	*138.66s*	*0.0169*	*0.62*	*0.62*	*0.62*	*0.62*
Skill-by-Skills	*550.95s*	*0.0675*	*0.7483*	*0.7508*	*0.7433*	*0.7471*

Table 15: *Performance Comparison of Bag-of-Skills and Skill-by-Skill Approaches: This table summarizes the key metrics for both approaches, highlighting the significant performance improvements achieved by the Skill-by-Skill method.*

D.2.2 Hyperparameter for Skill-by-Skill Similarity Calculation

In this iteration, we evaluate the impact of the threshold hyperparameter in the Skill-by-Skill similarity calculation. The threshold determines the minimum cosine similarity score required for two skills to be considered a match. Four threshold values (0.3, 0.4, 0.5, and 0.6) were tested to analyse their effect on model performance, class separation, and processing time.

Results Threshold 0.3:

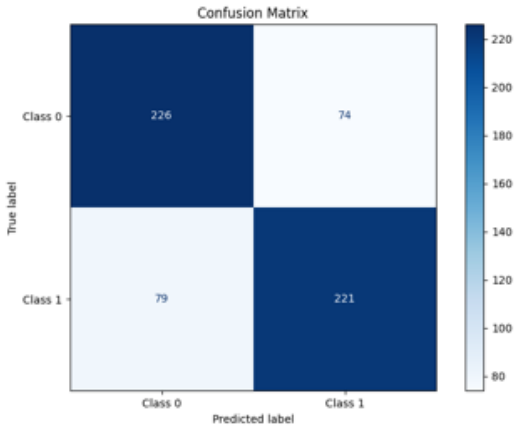


Figure 40: *Confusion Matrix - Threshold 0.3*

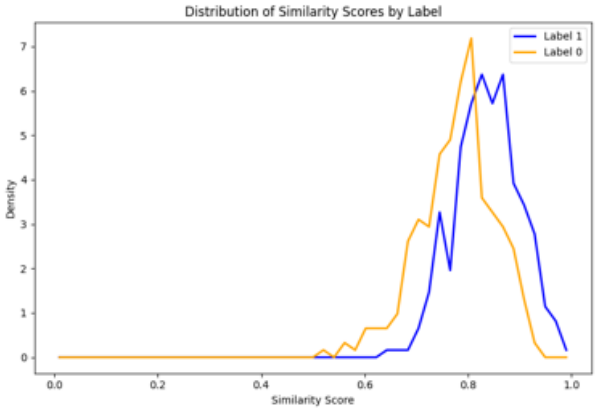


Figure 41: *Distribution of Similarity Score by Labels - Threshold 0.3*

Results Threshold 0.4:

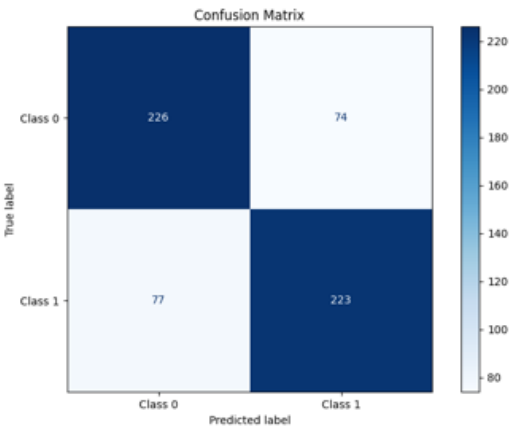


Figure 42: *Confusion-Matrix - Threshold 0.4*

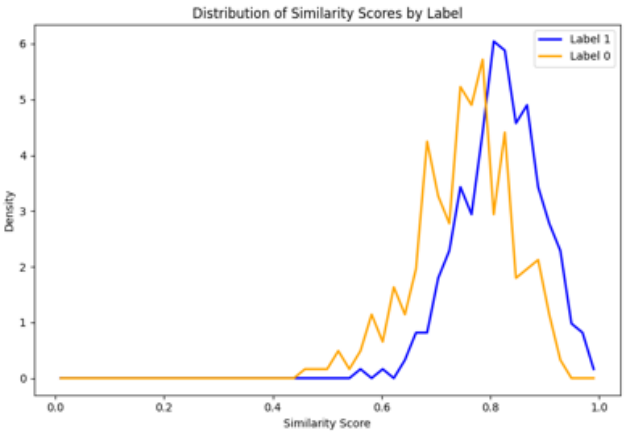


Figure 43: *Distribution of Similarity Score by Labels - Threshold 0.4*

#Results Threshold 0.5:

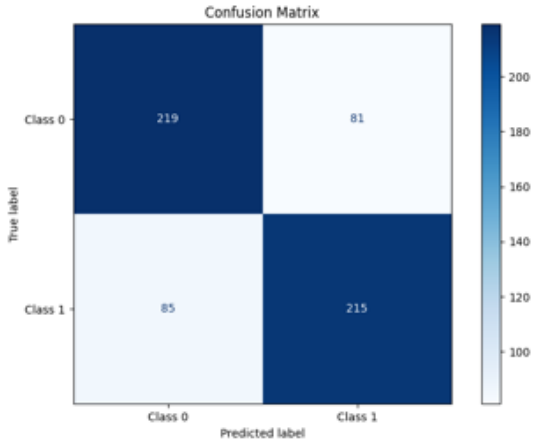


Figure 44: *Confusion-Matrix - Threshold 0.5*

Figure 45: *Distribution of Similarity Score by Labels - Threshold 0.5*

Results Threshold 0.6:

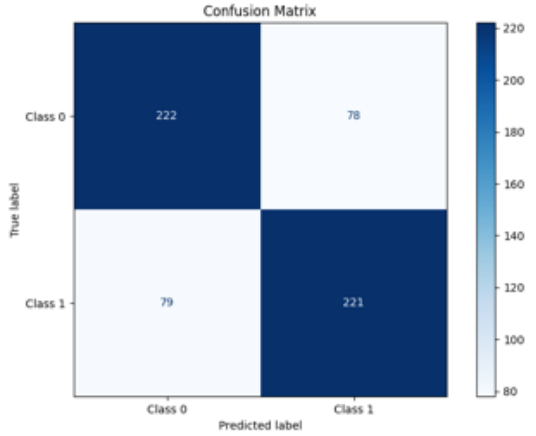


Figure 46: *Confusion-Matrix - Threshold 0.6*

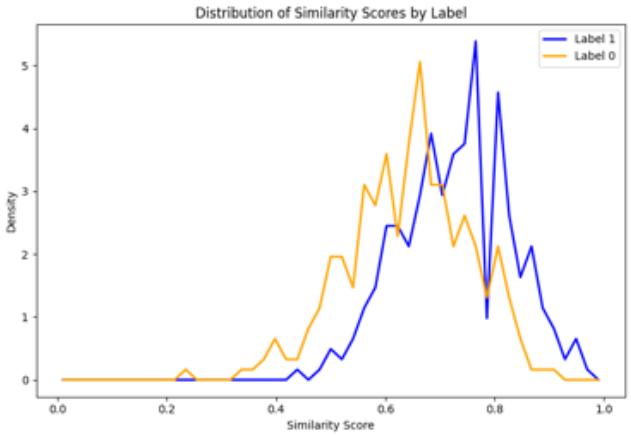


Figure 47: *Distribution of Similarity Score by Labels - Threshold 0.6*

The results show that a threshold of 0.4 achieved the best overall performance, yielding an F1-Score of 0.7471. This setting balanced precision and recall while maintaining a reasonable processing time of 550.95 seconds and achieving a Mean Similarity Score (SC) Difference of 0.0675, indicating strong class separation between good-fit (Label 1) and bad-fit (Label 0) résumés. Higher thresholds, such as 0.6, resulted in a slightly larger Mean SC Difference (0.0861), suggesting stricter matching criteria with better separation. However, this came at the cost of a lower F1-Score, as stricter thresholds excluded some good-fit résumés, reducing overall predictive performance. These findings validate the threshold of 0.4 as the optimal

choice, striking a balance between class separation, computational efficiency, and predictive performance.

Threshold	*Processing Time*	*Mean SC-Diff.*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
0.3	*658.04s*	*0.0572*	*0.745*	*0.7492*	*0.7367*	*0.7429*
0.4	*550.95s*	*0.0675*	*0.7483*	*0.7508*	*0.7433*	*0.7471*
0.5	*534.57s*	*0.0736*	*0.7233*	*0.7264*	*0.7167*	*0.7215*
0.6	*500.46s*	*0.0861*	*0.7383*	*0.7391*	*0.7367*	*0.7379*

Table 16: *The table compares the metrics among all thresholds. Results demonstrate that a threshold of 0.4 achieves the best balance, delivering the highest F1-Score (0.7471) and strong class separation, while maintaining reasonable processing time.*

D.2.3. Embedding Model

This section provides a detailed analysis of the embedding model evaluation, presenting the results of all-mpnet-base-v2, all-MiniLM-L6-v2, NV-Embed-v2, and SFR-Embedding. The comparison highlights critical performance metrics, including processing time, F1-Score, and Mean Similarity Score Difference, to demonstrate the trade-offs between performance and operational efficiency.

Embedding Model	*Model Size (Million Parameters)*	*Embedding Dimension*	*Memory Usage (GB, fp32)*
all-mpnet-base-v2	*110*	*768*	*0.41*
all-MiniLM-L6-v2	*33*	*384*	*0.12*
NV-Embed-v2	*7851*	*4096*	*29.25*
SFR-Embedding	*7111*	*4096*	*26.49*

Table 17: *Embedding Model Characteristics: This table shows the size, embedding dimensions, and memory usage of the evaluated models. Larger models like NV-Embed-v2 and SFR-Embedding demand significantly more resources, while all-MiniLM-L6-v2 is the most efficient option, balancing resource usage and scalability.*

Embedding Model	*Processing Time*	*Mean SC-Diff.*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
all-mpnet-base-v2	*140.80s*	*0.0631*	*0.6750*	*0.6780*	*0.6667*	*0.6723*
all-MiniLM-L6-v2	*106.30s*	*0.0623*	*0.6917*	*0.7018*	*0.6667*	*0.6838*
NV-Embed-v2	*5030.74s*	*0.0354*	*0.6917*	*0.6949*	*0.6833*	*0.6891*
SFR-Embedding	*5306.20s*	*0.0157*	*0.6667*	*0.6667*	*0.6667*	*0.6667*

Table 18: *Embedding Model Results: This table compares the performance metrics of the embedding models. all-MiniLM-L6-v2 delivers the best balance of speed and accuracy, while NV-Embed-v2 achieves the highest F1-Score but with excessive processing time, making it impractical for real-world use of the proposed model.*

Key Insights:

- all-MiniLM-L6-v2 emerges as the optimal choice, offering the shortest processing time, strong class separation, and reliable accuracy, making it ideal for scalable recruitment operations.
- While NV-Embed-v2 achieves the highest F1-Score, its impractical runtime and computational requirements limit its real-world applicability.
- The baseline model (all-mpnet-base-v2) remains a strong alternative but is slightly slower than all-MiniLM-L6-v2.
- SFR-Embedding underperforms in both accuracy and processing efficiency, making it the least suitable option.

Results all-mpnet-base-v2:

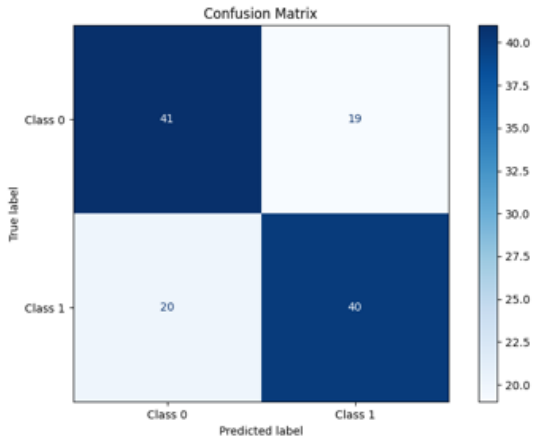


Figure 48: *Confusion-Matrix - all-mpnet-base-v2*

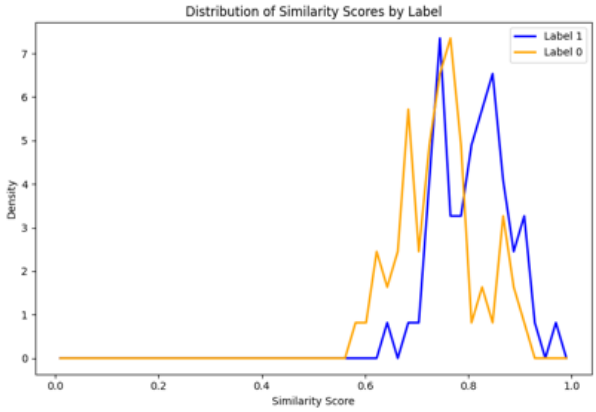


Figure 49: *all-mpnet-base-v2 - Distribution of Similarity Score by Labels*

Results all-MiniLM-L6-v2:

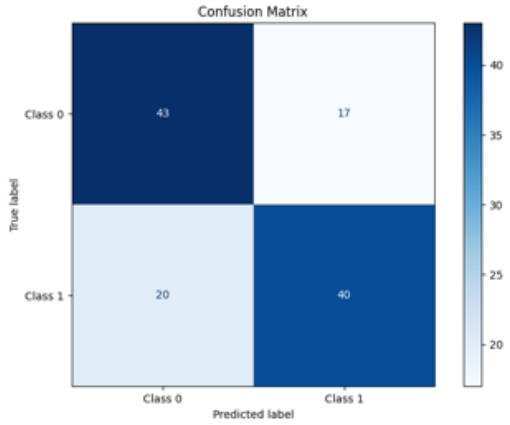


Figure 50: *all-MiniLM-L6-v2- Confusion-Matrix*

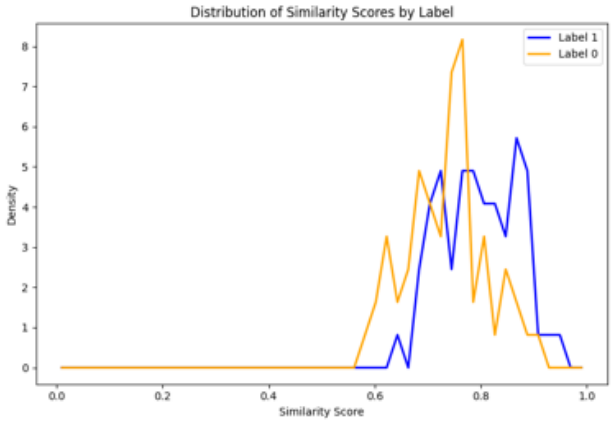


Figure 51: *all-MiniLM-L6-v2- Distribution of Similarity Score by Labels*

Results NV-Embed-v2:

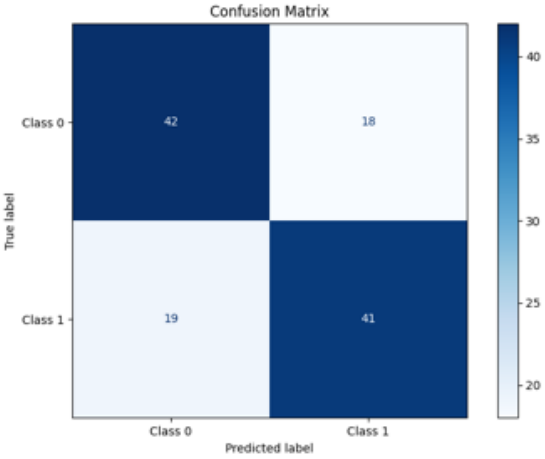


Figure 52: *NV-Embed-v2 - Confusion-Matrix*

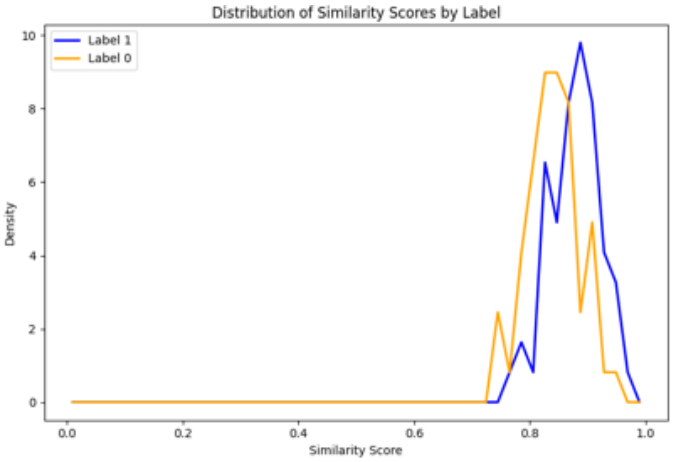


Figure 53: *NV-Embed-v2 - Distribution of Similarity Score by Labels*

Results SFR-Embedding-Mistral:

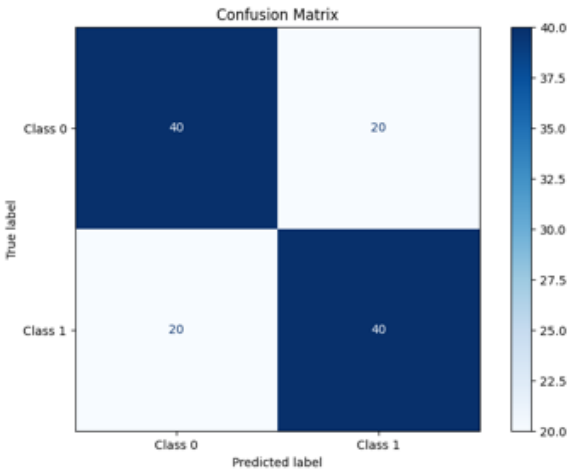


Figure 54: *SFR-Embedding-Mistral – Confusion Matrix*

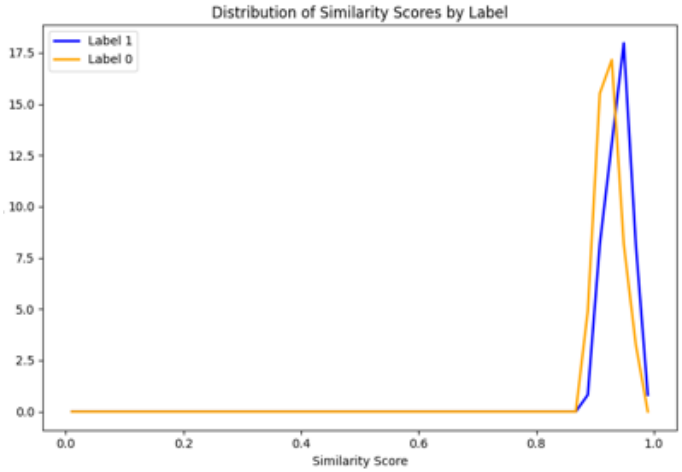


Figure 55: *SFR-Embedding-Mistral - Distribution of Similarity Score by Labels*

D.2.4 Transformer Architectures

The baseline model employs a bi-encoder (figure x) architecture, independently encoding inputs (résumé and job description skills) into fixed-size embeddings for similarity comparison. In contrast, cross-encoders jointly process input pairs, expected to catch richer interactions between inputs (see 5.2.6 for details). While cross-encoders have shown superior performance on certain tasks (Reimers and Gurevych 2019), their computational demands are significantly higher. This iteration evaluates the two architectures, comparing their computational efficiency and performance outcomes under testing conditions.

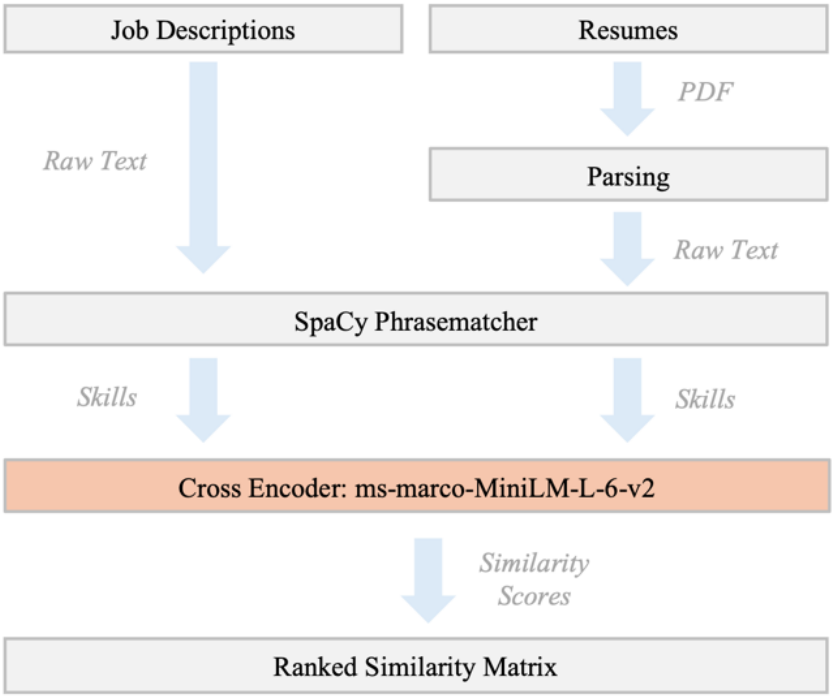


Figure 56: Cross-Encoder Workflow: *This diagram illustrates the Cross-Encoder architecture's workflow. Unlike the Bi-Encoder, the Cross-Encoder jointly processes job descriptions and résumés, generating a similarity score in a single step.*

This section provides an in-depth evaluation of the Bi-Encoder and Cross-Encoder architectures, comparing their performance across key metrics, including F1-Score, and Processing Time. The analysis highlights the trade-offs between computational efficiency and class separation quality. While the Bi-Encoder efficiently processes résumés and job descriptions by generating independent embeddings, the Cross-Encoder processes input pairs jointly, theoretically capturing richer interactions. However, as the results indicate, the Cross-Encoder's higher computational cost does not translate into significantly better class separation or performance.

Modelling Approach	*Processing Time*	*Mean SC-Diff.*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Bi-Encoder	*749.35s*	*0.0691*	*0.7433*	*0.7483*	*0.7333*	*0.7407*
Cross-Encoder	*4342.99s*	*0.1337*	*0.74*	*0.74*	*0.74*	*0.74*

Table 19: Performance Comparison of Bi-Encoder and Cross-Encoder *- This table compares the performance of the Bi-Encoder and Cross-Encoder architectures. While both achieve nearly similar F1-Scores, the* ***Bi-Encoder*** *significantly outperforms the Cross-Encoder in* ***processing time*** *(749.35s vs. 4342.99s)*

Appendix

The following visualizations and tables illustrate the findings, showcasing confusion matrices, similarity score distributions, and the final performance metrics for both architectures.

Results Bi-Encoder:

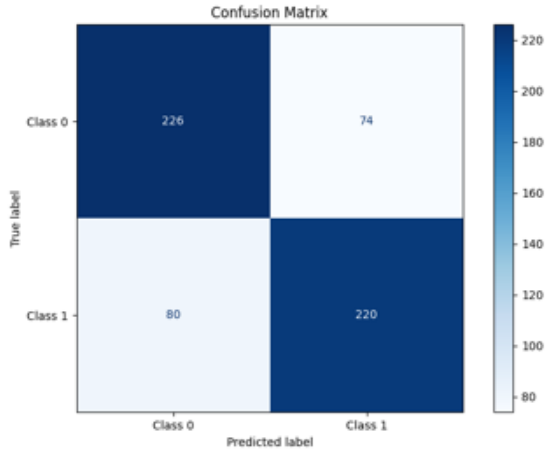


Figure 57: *Confusion-Matrix - Bi-Encoder*

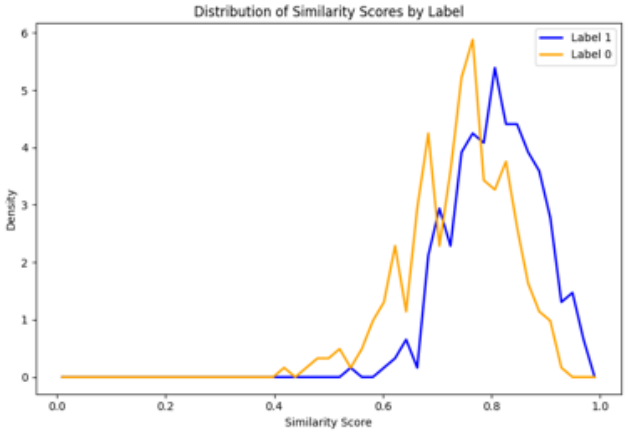


Figure 58: *Distribution of Similarity Score by Labels - Bi-Encoder*

Results Cross-Encoder:

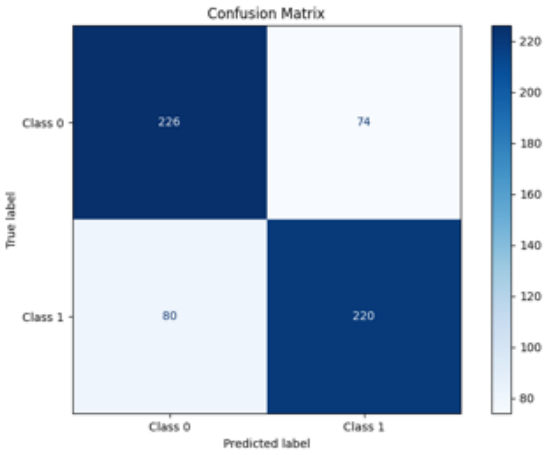


Figure 59: *Confusion-Matrix - Cross-Encoder*

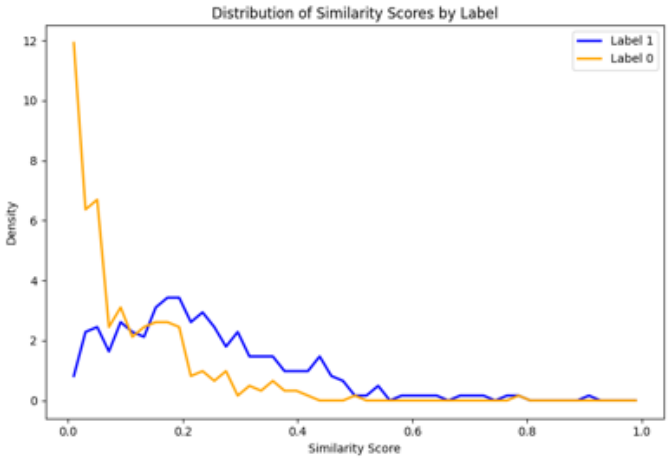


Figure 60: *Distribution of Similarity Score by Labels - Cross-Encoder*

Appendix

Analysis of Cross-Encoder Results:

While the Cross-Encoder architecture is theoretically expected to capture richer interactions between inputs by processing résumé and job description pairs jointly, the **similarity score distribution graph (Figure 54)** reveals its limitations. Unlike the Bi-Encoder, which demonstrates clear separation between good-fit (Label 1) and bad-fit (Label 0) résumés, the Cross-Encoder produces overlapping similarity scores across the two classes.

Reasons for Poor Performance in Similarity Distribution

1. **Overfitting to Training Data**: The Cross-Encoder's joint processing may overfit specific patterns in the synthetic dataset, reducing its ability to generalize and distinguish between good and bad fits.
2. **Lack of Granular Skill Separation**: The richer contextual relationships captured by the Cross-Encoder may dilute the signal for individual skill matching, leading to ambiguity in score distinctions.
3. **Data Limitations**: The synthetic dataset, while controlled, may not provide sufficient variability for the Cross-Encoder to leverage its full potential, causing it to underperform in distinguishing nuanced differences.
4. **Computational Instability**: The significant computational load of the Cross-Encoder may impact performance stability, particularly when processing larger datasets, resulting in inconsistent outputs.

Implications

The Cross-Encoder's inability to achieve clear class separation, combined with its excessive **processing time (4342.99 seconds)**, limits its practical applicability for real-world recruitment

systems. These findings further reinforce the **Bi-Encoder** as the preferred architecture, offering a strong balance of performance, clarity in class separation, and operational efficiency.

Appendix E – Skills-Based Similarity

This appendix describes the datasets and processes used to develop the skills-based similarity model presented in Chapter 8. It explains structure, cleaning and filtering of Technology Skills and the Occupation Data datasets from O*NET. It gives further explanations on how role vectors are created and how additional features are included. At the end, the code for building the model is provided, showing the steps for embedding skills with MPNET and calculating cosine similarity to rank roles.

E.1 Exploring O*NET databases and Data Cleaning

This section provides some insight about data structure and cleaning, to offer a more complete overview of the information used in this study.

Technology Skills Dataset consists of **7 columns** and multiple rows, organized as follows:

1. **O*NET-SOC Code**: The standardized code for each occupation.
2. **Title**: The job title associated with the role.
3. **Example**: A specific example of software or technology relevant to the role.
4. **Commodity Code**: A unique code identifying the category of the technology or software.
5. **Commodity Title**: The category description for the technology or software.
6. **Hot Technology**: Indicates whether the technology is classified as critical for that specific role (Y/N).
7. **In Demand**: Indicates if the role or technology is currently in high demand across all job postings (Y/N).

*O*NET-SOC Code*	*Title*	*Example*	*Commodity Code*	*Commodity Title*	*Hot Technology*	*In Demand*
11-1011.00	*Chief Executives*	*Adobe Acrobat*	*43232202*	*Document management software*	*Y*	*N*
11-1011.00	*Chief Executives*	*Adobe Acrobat*	*43232306*	*Data base user interface and query software*	*N*	*N*
11-1011.00	*Chief Executives*	*Adobe Acrobat*	*43232201*	*Content workflow software*	*Y*	*Y*
11-1011.00	*Chief Executives*	*Blackbaud The Raiser's Edge*	*43232303*	*Customer relationship management CRM software*	*N*	*Y*

Table 20: *Example of how the Technology Skills dataset looks like*

Since the *Commodity Code* and *Commodity Title* fields were not used, they were removed for simplicity. Additionally, *Structured Query Language (SQL)* was standardized to *SQL* for consistency. *Microsoft Office Software* was excluded as it was too generic and redundant, given that specific Microsoft tools like *Excel* and *PowerPoint* were already included in the dataset.

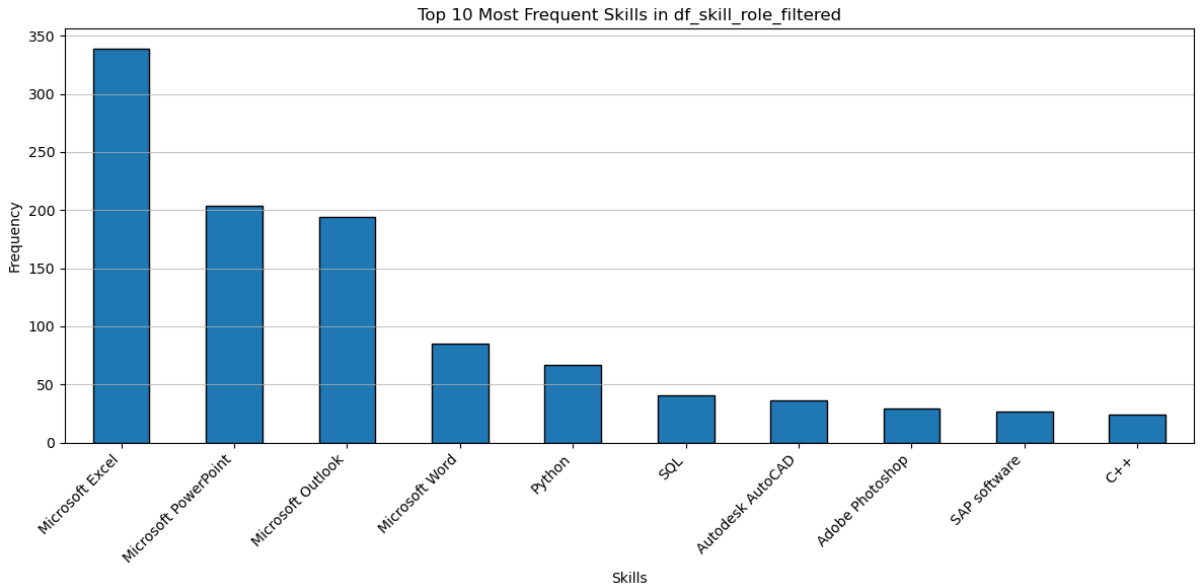


Figure 61: Frequency distribution of top 10 Hard Skills more mentioned in the dataset: *These tools are widely used across various roles, regardless of industry, which is unsurprising given their long-standing presence in the workplace. Additionally, the presence of programming languages such as Python and SQL indicates their growing importance across different roles. This trend reflects the increasing demand for data-related and technical skills in modern job markets, suggesting that Python is becoming essential in more industries beyond traditional tech fields*

To better understand how roles are represented, the skills were filtered to include only those classified as *In-Demand* and *Hot Technologies*. These filtered skills were then grouped by role,

resulting in a table with two columns: one for the *Role Titles* and the other for the corresponding list of *Hard Skills* associated with each role. The resulting table looks like the one below:

Title	***Skills***
Logistic Analysts	*[Microsoft Access', 'Microsoft Excel', 'Microsoft Outlook', 'Microsoft Power BI', 'Microsoft PowerPoint', 'SAP software', 'SQL', 'Tableau']*
Market Research Analysts	*['Google Analytics', 'Microsoft Excel', 'Microsoft Outlook', 'Microsoft PowerPoint', 'Salesforce software']*
Sales Engineers	*[''Amazon Web Services AWS software', 'Linux', 'Microsoft Azure software', 'Microsoft Excel', 'Microsoft Outlook', 'Microsoft PowerPoint', 'Python', 'Salesforce software']*
Statisticians	*['IBM SPSS Statistics', 'Microsoft Access', 'Microsoft Excel', 'Microsoft PowerPoint', 'Python', 'R', 'SAS', 'SQL', 'Tableau']*

Table 21: Hard Skills per Role: *The table provides an overview of hard skills in- demand per each role collected from O*NET*

Occupation Data Skills Dataset consists of **3 columns** and multiple rows, organized as follows:

1. **O*NET-SOC Code**: The standardized code assigned to each occupation in the O*NET classification system.
2. **Title**: The occupational title corresponding to the O*NET-SOC Code.
3. **Description**: A detailed description of the role's primary responsibilities and duties.

O*NET-SOC Code	***Title***	***Description***
11-2011.00	*Advertising and Promotions Managers*	*Plan, direct, or coordinate advertising policies and programs or produce collateral materials, such as posters, contests, coupons, or giveaways, to create extra interest in the purchase of a product or service for a department, organization, or account basis.*
11-2022.00	*Marketing Managers*	*Plan, direct, or coordinate marketing policies and programs, such as determining the demand for products and services offered by a firm and its competitors, and identify potential customers. Develop pricing strategies to maximize profits and satisfy customers. Oversee product development or monitor trends that indicate the need for new products and services.*

Table 22: *Example of how the Occupation Data looks like*

To extract keywords from the role descriptions, the function *main_branch.job_info_extraction* was applied to the relevant column. This function, located in the *Services/sJobInfoExtraction.py* directory of our codebase, is based on the process explained in Chapter 5. By applying this function, we generated a set of contextual keywords for each role, helping to better define the nature and scope of each position.

Title	***Skills***
Mechanical Engineers	*['engineering']*
Market Research Analysts	*['search engine', 'marketing', 'advertising', 'operations research"]*
Computer and Information Systems Managers	*['computer programming', 'data processing"]*

Table 23: Keywords from role descriptions: *The table provides an overview of extracted keywords per each role collected from O*NET*

The Occupation Data file includes 1016 roles, while the Technology Skills file contains only 453 roles. Consequently, the analysis focuses on the 453 roles common to both datasets.

E.2 Skills-Based Model Steps

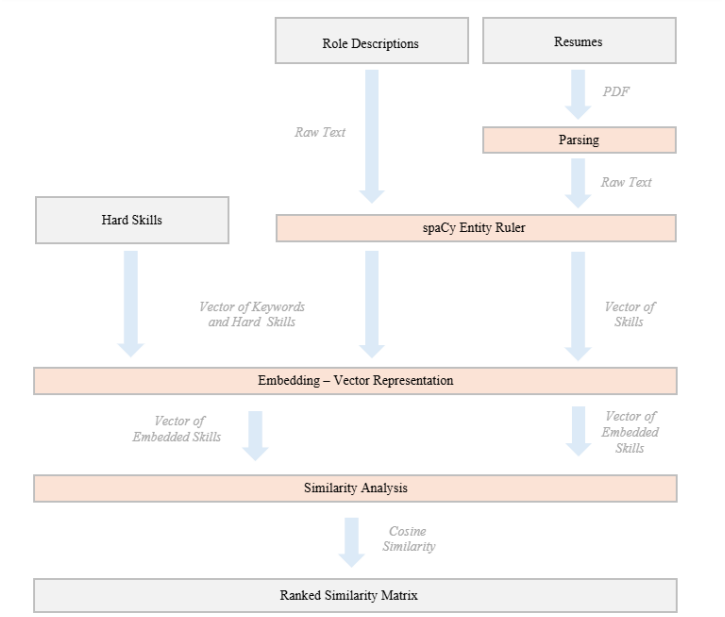


Figure 62: Skills Based Model Steps. *First, skills from résumés in PDF format are extracted with the spaCy Entity Ruler. Role Vectors are created combining extract skills from role descriptions and hard skills from O*NET. We then embed the skills using a semantic embedding model (in our case MPNET) before we conduct a similarity analysis with cosine similarity.*

E.3 Additional Features

Since O*NET has gathered extensive information about the job market and emerging trends, additional data was included to provide users with a more comprehensive overview of roles suitable for potential transitions.

To enhance this functionality, information on average annual wages from the May 2023 National Occupational Employment and Wage Estimates (Bureau of Labor Statistics 2023) was integrated. In this way, the tool can indicate how a person's salary might increase or decrease if they transition into a particular role. This information is based on U.S. data, and while it provides valuable insights, it is important to note that it may not reflect variations across countries or regions. With more time and resources, a dynamic system could be developed to ensure this data remains up to date and customizable based on the user's location.

To provide insights into potential salary changes when transitioning to a different role, users are required to input their current annual salary in US dollars (the currency used by the Bureau of Labor Statistics) into the model. Since both O*NET and the Bureau of Labor Statistics utilize the Standard Occupational Classification (SOC) system to identify roles, this alignment enables seamless integration of salary data. The resulting table combines role titles, associated skills, and the annual mean wage, offering users a clearer understanding of how their salary might increase or decrease based on a potential career transition.

Title	***Skills***	***Annual mean wage ($)***
Mechanical Engineers	*['Autodesk AutoCAD', 'Autodesk Revit', 'Microsoft PowerPoint', 'Microsoft Excel', 'Python', 'Dassault Systems SolidWorks', 'Microsoft Outlook', 'engineering']*	*105,220*
Market Research Analysts	*['Google Analytics', 'Microsoft PowerPoint', 'search engine', 'Microsoft Excel', 'marketing', 'advertising', 'operations research', 'Microsoft Outlook', 'Salesforce software']*	*83,190*
Computer and Information Systems Managers	*['SQL', 'Microsoft Azure software', 'computer programming', 'Amazon Web Services AWS software', 'data processing', 'Microsoft Excel', 'SAP software']*	*180,720*

Table 24: *Table after adding information about annual mean salary: The table show how information about wages is included to allow the model to provide more infomration about the roles and giving back how much the salary differs compared to the sctual one of the user.*

Additionally, O*NET's All Bright Outlook Occupations dataset provides a classification of roles based on factors such as projected growth, the number of job openings, and whether they are emerging occupations. This data gives users critical insights, allowing them to make informed decisions about career transitions. This table is directly used as input to the model, and the information is mapped to each role.

Together, these additions aim to provide a well-rounded perspective, enabling users to prioritize roles that are not only compatible with their skills but also align with future job market demands and salary expectations.

E.4 Function for Final Output

Note: To enhance usability, the "Sample of Reported Titles" dataset expands the list of job titles users can choose from, helping them find a role that closely matches their current job. The user can choose “Brand Manager”, instead of “Marketing Manager”.

Because the analysis uses general role titles, the selected title is then mapped to a corresponding generic role, which is used as input for the similarity calculation function. This approach ensures that the tool remains user-friendly while offering a comprehensive and personalized analysis.

Code 7: *The similarity_final_3 function computes cosine similarity between a single CV and multiple job descriptions using precomputed MPNET embeddings of skills. The function integrates role-to-role similarity scores, missing skills, growth potential, and salary variation to enhance candidate recommendations.*

```
def similarity_final_3(cv_df, job_df=merged_df, hard_skills=df_skill_role_grouped, scores_df=similarity_df,
titles_df=titles_df, growth_df=growth_df, role=None, wage=0, parallel=False):
    """
    Calculate cosine similarity between a single CV and multiple job descriptions.
    Optionally filter based on a specific role and included role-specific scores, missing skills,
    growth potential, and salary variation in the output.

    Parameters:
    - cv_df (DataFrame): DataFrame containing the CV's skills.
    - job_df (DataFrame): DataFrame containing vecotrs defining roles and salary information(default is merged_df).
    - hard_skills (DataFrame): DataFrame containing hard skills for each role.
    - scores_df (DataFrame): DataFrame containing precomputed role-specific scores based on similarity between pairs
of roles.
    - titles_df (DataFrame): DataFrame mapping more specific job titles to generic roles.
    - growth_df (DataFrame): DataFrame containing information about growth potential, number of job posting, and in-
demand information for roles.
    - role (str): The specific role to filter by.
    - wage (float): The current wage to calculate salary variation.
```

```
    - parallel (bool): Whether to use parallel processing for embedding computation.

    Returns:
    - ranked_jobs (DataFrame): DataFrame with similarity scores and rankings.
    - ranked_jobs_2 (DataFrame): DataFrame with role-specific scores and rankings.
    """

    # Initialize the MPNET model for embedding
    model = SentenceTransformer('all-mpnet-base-v2')
    model.max_seq_length = 75
    model.tokenizer.padding_side = "right"
    model.eval()

    def add_eos(input_examples):
        """Helper function to add end-of-sequence tokens to each skill."""
        input_examples = [input_example + model.tokenizer.eos_token for input_example in input_examples]
        return input_examples

    def missing_skills(skills, cv):
        """Identify skills from the job description that are missing in the CV."""
        skill_words = set(skill.lower().strip() for skill in skills if isinstance(skill, str))
        cv_set = set(cv.lower().strip() for cv in cv if isinstance(cv, str))
        missing_words = skill_words - cv_set
        return list(missing_words)

    # Find the generic title corresponding to the input role
    generic_title = titles_df.loc[titles_df['Reported Job Title'] == role, 'Title'].iloc[0]

    # Filter the scores_df using the generic title
    filtered_scores_df = scores_df[scores_df['Target_Role'] == generic_title]

    # Create a dictionary to map role-specific scores
    role_score_mapping = dict(zip(filtered_scores_df['Role'], filtered_scores_df['composite_tasks_dwas_ksas']))

    # Extract and process the CV's skills
    cv_df['Skills_Text'] = cv_df['Skills'].apply(add_eos)
    cv_df['Skills_Text'] = cv_df['Skills_Text'].apply(
        lambda x: ' '.join(sorted(set(x))) if isinstance(x, list) else ''
    )

    # Generate embedding for the CV
    cv_embedding = model.encode(
        cv_df['Skills_Text'].iloc[0],  # Assuming a single CV is provided
        batch_size=1,
        show_progress_bar=False
    )

    # Process and encode the job descriptions' skills
    job_df['Skills_Text'] = job_df['Skills'].apply(add_eos)
    job_df['Skills_Text'] = job_df['Skills_Text'].apply(
        lambda x: ' '.join(sorted(set(x))) if isinstance(x, list) else ''
    )

    job_embeddings = model.encode(
        job_df['Skills_Text'].tolist(),
        batch_size=32,
        num_workers=os.cpu_count() // 2 if parallel else 0,
        show_progress_bar=False
    )

    # Compute cosine similarity between the CV and job embeddings
    similarity_scores = cosine_similarity([cv_embedding], job_embeddings).flatten()
    job_df['similarity_score'] = similarity_scores

    # Normalize similarity scores using z-score, since also scores from precomputed role-specific scores are
normalized the same way
    mean_score = job_df['similarity_score'].mean()
    std_score = job_df['similarity_score'].std()
    if std_score > 0:
        job_df['normalized_similarity_score'] = (job_df['similarity_score'] - mean_score) / std_score
    else:
        job_df['normalized_similarity_score'] = 0  # Assign 0 if std is zero (all scores are identical)

    # Rank the jobs by normalized similarity score
    job_df['rank'] = job_df['normalized_similarity_score'].rank(ascending=False)

    # Add missing skills for each job description
    job_df['missing_skills'] = job_df.apply(
        lambda row: missing_skills(
            hard_skills.loc[row.name, 'Skills'],  # Skills from the hard_skills DataFrame
            cv_df['Skills'].iloc[0]  # Skills from the first row of cv_df
        ),
        axis=1
```

```
    )

    # Map growth potential information to each role
    role_growth_mapping = dict(zip(growth_df['Occupation'], growth_df['Categories']))
    job_df['role_growth'] = job_df['Title'].map(role_growth_mapping)

    # Fill NaN values in 'role_growth' with 'Not In-Demand'
    job_df['role_growth'].fillna('Not In-Demand', inplace=True)

    # Calculate annual wage variation or mark as "Info Not Available" if wage is missing
    job_df['annual_wage_variation'] = job_df.apply(
        lambda row: "Info Not Available" if row['annual_wage'] == 0 else row['annual_wage'] - wage,
        axis=1
    )

    # Create a copy of job_df for role-to-role similarity scores
    job_df_2 = job_df.copy()

    # Add role-specific scores to a new column in job_df_2
    job_df_2['role_scores'] = job_df_2['Title'].apply(lambda x: role_score_mapping.get(x, None))

    # Rank the jobs by role-to-role similarity scores
    job_df_2['rank'] = job_df_2['role_scores'].rank(ascending=False)

    # Prepare final DataFrames with relevant columns
    ranked_jobs_to_role = job_df_2[['rank', 'Title', 'missing_skills', 'annual_wage_variation',
'role_growth']].sort_values(by='rank', ascending=True)
    ranked_jobs = job_df[['rank', 'Title', 'missing_skills', 'annual_wage_variation',
'role_growth']].sort_values(by='rank', ascending=True)

    # Return the two DataFrames
    return ranked_jobs, ranked_jobs_to_role
```

The function generates two tables displaying the top 20 most similar roles. The first table is based on the candidate's résumés and ranks roles using cosine similarity, primarily focusing on hard skills. The second table allows users to select their closest role from a list displayed in the web application and ranks roles using a score that incorporates additional factors such as knowledge, abilities, values, and other relevant criteria. Both tables include information on salary variations, calculated based on the wage input provided by the user, as well as projections on the expected growth of each role in the coming years. This dual-table approach provides an overview to guide career transitions.

Appendix F - Role Recommendations

F.1 Role Similarity Calculation

Scale Definitions From O*NET

Importance reflects how important a feature (e.g. skill, knowledge, ability) is to successfully perform an occupation, rated from 1 ("Not Important") to 5 ("Extremely Important"). Level indicates the degree of proficiency required, rated from 0 to 7, offering insight into the intensity needed (National Center for O*NET Development 2024).

Precision Issues Related to Level Values

4,963 level values were flagged as “Recommended Suppression” for low precision, compared to only 19 importance values.

Squared Euclidian Distance

The Squared Euclidean Distance between two vectors $x = (x_1, x_2, \ldots., x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ is given by:

$d^2(x, y) = \sum_{i=1}^{n}(x_i - y_i)^2$

This distance measures the sum of the squared differences between corresponding elements in two vectors (Dokmanic, et. al 2015). It is a variation of the standard Euclidean Distance, but without taking the square root. This makes it computationally efficient for large datasets and suitable for tasks where the magnitude of differences matters.

Tasks Scale Definitions

Tasks were filtered based on their importance category, removing supplemental tasks.

Core — tasks that are critical to the occupation. The criteria for these tasks are (a) relevance ≥ 67% and (b) a mean importance rating of ≥ 3.0.

Supplemental — tasks that are less relevant and/or important to the occupation. Two sets of tasks are included in this category: (a) tasks rated ≥ 67% on relevance but < 3.0 on importance,

and (b) tasks rated < 67% on relevance, regardless of mean importance (National Center for ONET Development 2024).

F.2 Feature Selection

Pearsons Correlation Coefficient

For work values, work styles and work interest, similarities were used assessing Pearson correlation coefficient which measures the strength and direction of the linear relationship between two variables x and y. It ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no correlation. The formula calculates the ratio of the covariance between the variables to the product of their standard deviations and is given by (Cohen, Huang and Benesty, 2009):

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where:

- r = Pearson's correlation coefficient
- x_i and y_i = Individual data points in the datasets x and y
- $\bar{x}$ and $\bar{y}$: The means of x and y
- n: The number of data points

Descriptive Statistics of Criteria

Criteria	count	mean	std	min	25%	50%	75%	max
Title Similarity	1031240	0,28	0,13	-0,13	0,19	0,27	0,36	0,96
Styles Correlation	772641	0,49	0,26	-0,75	0,35	0,54	0,69	1
Values Correlation	763876	0,06	0,49	-1	-0,33	0,05	0,47	1
Values Correlation	851929	0,21	0,43	-0,97	-0,09	0,18	0,5	1
Salary Difference	247456	31415,31	30903,47	0	9690	22120	43490	204830

Table 25: Descriptive Statistics of Criterion: *This figure presents the descriptive statistics for each evaluation criterion, including count, mean, standard deviation (std), minimum (min), and key percentiles (25%, 50%, 75%), as well as the maximum (max) values.*

Appendix

Intercorrelations

Intercorrelations were calculated by finding the intercorrelation for each role pair and then averaging the results. The observations indicate that these measures describe different information about the similarities between occupations.

Criterion	Criterion	Mean Correlation	Std	Min	Max
Interests	Median Salary	-0,09	0,16	-1,00	0,68
Interests	Styles	0,14	0,16	-0,91	1,00
Interests	Title	0,24	0,19	-0,46	1,00
Interests	Values	0,10	0,19	-1,00	0,86
Median Salary	Styles	-0,10	0,15	-1,00	0,59
Median Salary	Title	-0,07	0,17	-1,00	0,80
Median Salary	Values	-0,17	0,24	-0,91	1,00
Styles	Title	0,14	0,16	-0,62	1,00
Styles	Values	0,05	0,13	-1,00	0,65
Title	Values	0,11	0,17	-1,00	0,68

Table 26: Intercorrelations between Criteria*: Intercorrelations were calculated by finding the intercorrelation for each role pair and then averaging the results. The observations indicate that these measures describe different information about the similarities between occupations.*

Justification of Similarity Measures for Work Values, Styles and Interests

Since work values, work styles, and work interests are inherently abstract and conceptual, importance differences are considered less meaningful than the relative rank-ordering of the criteria (Dahlke, et. al 2022). As a result, similarities were assessed using Pearson correlation coefficients, which focus on how attributes are prioritized rather than their absolute values. For example, the ordered preferences for work values (e.g., ranking "achievement" higher than "autonomy") were seen as more useful for distinguishing occupations than comparing whether occupations score higher or lower in absolute importance terms.

Correlation Analysis Between Features and Criteria

For each target role, the similarity scores for each feature and criterion were correlated with those of all other roles. These correlations were then averaged across all target roles, providing an overall estimate of how well each pair of similarity metrics aligns in terms of occupational rank ordering for a given criterion.

Criterion	Predictor	Correlation	Standard Deviation
Role Title	KSAs	0,366	0,0002
Role Title	Tasks	0,509	0,0002
Role Title	DWAs	0,449	0,0002
Work Values	KSAs	0,361	0,0003
Work Values	Tasks	0,223	0,0002
Work Values	DWAs	0,239	0,0002
Work Styles	KSAs	0,282	0,0002
Work Styles	Tasks	0,208	0,0003
Work Styles	DWAs	0,220	0,0003
Work Interests	KSAs	0,439	0,0003
Work Interests	Tasks	0,367	0,0003
Work Interests	DWAs	0,407	0,0003
Salary	KSAs	-0,312	0,0007
Salary	Tasks	-0,088	0,0006
Salary	DWAs	-0,126	0,0007

Table 27: Correlation Analysis between Features and Predictors*: This table displays the correlation values and standard deviations of different predictors (e.g., KSAs, Tasks, and DWAs) for each evaluation criterion. It provides insight into how strongly each predictor aligns with specific criteria and the variability within these measures, supporting the analysis of predictor effectiveness for role similarity.*

Regression Equations

Mark that for each model, there were conducted 5 regression analysis for each criterion: Work Values, Work Styles, Work Interest and Median Salary

Model 1: KSAs Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1 X_{KSA} + \epsilon,$$

Criterion = Work values, work styles, work interest, median salary

Model 2: Tasks Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1 X_{Tasks} + \epsilon,$$

Criterion = Work values, work styles, work interest, median salary

Model 3: Activities Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1 X_{DWA} + \epsilon,$$

Criterion = Work values, work styles, work interest, median salary

Appendix

Model 4: Tasks & Activities Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1\, X_{Tasks} + \beta 2 X_{DWA} + \epsilon,$$

Criterion = Work values, work styles, work interest, median salary

Model 5: Tasks & KSAs Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1\, X_{Tasks} + \beta 2 X_{KSA} + \epsilon,$$

Model 6: Activities & KSAs Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1\, X_{KSA} + \beta 2 X_{DWA} + \epsilon,$$

Model 7: Activities, KSA & DWA Predicting Criteria

$$Y_{criterion} = \beta 0 + \beta 1\, X_{KSA} + \beta 2 X_{DWA} + \beta 3 X_{Tasks} + \epsilon,$$

R-squared Analysis

Job Title: Model 5 (Tasks + Activities) achieves the highest R^2 (0.264), followed closely by Model 2 (Tasks alone, R^2 = 0.259) and Model 7 (Tasks + Activities + KSAs, R^2 = 0.241). Tasks alone explain most of the variance, reinforcing their importance for aligning job titles.

Work Values: Model 1 (KSA) performs best (R^2 = 0.13), closely followed by Model 4 (Tasks + KSA) and Model 6 (KSA + Activities). KSAs are the dominant driver of work value similarity.

Work Styles: KSAs are the strongest individual predictor (Model 1, R^2 = 0.079), with only marginal improvements when combining Tasks and Activities (Model 7, R^2 = 0.086). The low adjusted R square value suggests that while KSAs are the most explanatory factor for styles similarity, these models struggle to capture much of the variance.

Work Interests: Work Interests show a slightly higher R^2 compared to other soft related variables. KSAs alone are the best individual predictor (Model 1, $R^2 = 0.192$), but the combined model (Model 7, $R^2 = 0.238$) provides the highest explanatory power. Adding Activities and Tasks enhances the model, indicating that Interests slightly benefit from a combination of all three features.

Salary: KSAs alone offer the highest but still weak explanation (Model 1, $R^2 = 0.097$). Adding Tasks and Activities (Model 7, $R^2 = 0.064$) decreases explanatory value, suggesting that salary differences arise from factors not captured by these features.

F.3 Benchmarking Features Against O*NET Rankings

Justification for method

Precision@K, Mean Reciprocal Rank and Mean Average Precision are rank similarity measures widely used in research for evaluating ranking quality, especially when the focus is on ensuring top-weighted relevance and handling incomplete rankings (Luchen and Clearke 2014; Konstantinidis, et. al 2022).

Precision@K

Precision@K measures the proportion of relevant items within the top K results of the ranked list. For example, in the context of our model, Precision@10 evaluates how many of the top 10 occupations recommended by the models are also listed as similar occupations in O*NET top 20 (Luchen and Clearke 2014).

$$\text{Precision@K} = \frac{\text{Number of relevant items in top } K}{K}$$

Mean Reciporcal Rank (MRR)

MRR evaluates how quickly the first relevant item appears in the ranked list by averaging the reciprocal of the rank position of the first relevant item across all queries (Luchen and Clearke 2014)

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}$$

Where:

- N is the number of queries
- rank_i is the rank position of the first relevant item for the i-th query.

Mean Average Precision (MAP)

MAP is calculated by averaging the precision scores at the ranks where relevant items appear, reflecting both precision and rank position (Luchen and Clearke 2014).

$$\text{MAP} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i}\sum_{j=1}^{m_i}\text{Precision@}r_{ij}$$

Where:

- N is the number of queries
- m_i is the number of relevant items for the i-th query
- r_{ij} is the rank position of the j-th relevant item for the i-th query.

F.4 Graphical Representation of the Model Output

Example Output of the Final Model

This network graph visualizes the top recommended occupations for "Accountants and Auditors," based on the role-to-role matching model. Each node represents a related occupation, and the edges display the similarity scores between the roles.

The closest matches, such as Financial Examiners (3.11), Budget Analysts (3.01), and Treasurers and Controllers (2.90), are highlighted with thicker lines, indicating stronger similarities. The recommendations demonstrate the model's effectiveness in identifying relevant and suitable career paths.

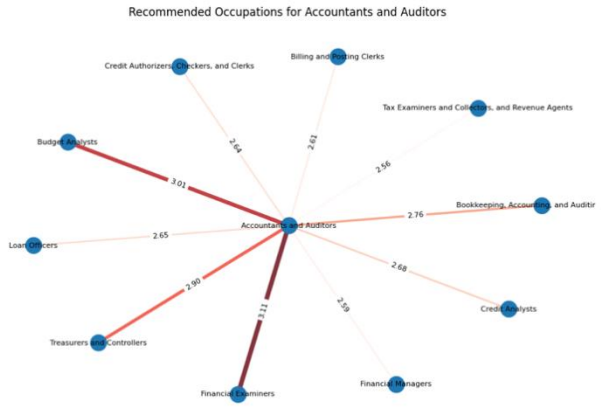


Figure 63: Output Example of Accountants and Auditors: *This network graph visualizes the top recommended occupations for "Accountants and Auditors," based on the role-to-role matching model.*

Community Analysis Methodology

The community analysis is performed by applying the Louvain algorithm from the NetworkX library (NetworkX Developers 2024) to detect and measure clusters within the filtered graph, where nodes represent occupations and edges indicate similarities. Louvain was chosen over other community algorithms such as InfoMap because it has proven to perform better (Emmons, et. al 2016).

Communities Characteristics Summarized

Community	N Roles	Example Roles	Top Occupation Keywords
0	272	Investment Fund Managers, Supervisors of Personal Service Workers	Managers (33), Clerks (23), Supervisors (18)
1	249	Structural Iron and Steel Workers, Civil Engineering Technologists	Technicians (42), Engineers (37), Repairers (28)
2	182	Psychology Teachers, Law Teachers, Sports Medicine Physicians	Teachers (46), Postsecondary (36), Education (18)
3	129	Shoe and Leather Workers , Waiters , Glass Blowers	Operators (41), Machine (35), Tenders (35)
4	24	Tree Trimmers and Pruners, Fishing and Hunting Workers	Operators (13), Equipment (5), Oil (4)
5	22	Mathematicians, Computer Programmers, Web Developers	Computer (8), Specialist (4), Analyst (4)

Table 28: Communities Characteristics: *This table summarizes the six identified occupational communities, listing the number of roles in each, example occupations, and key keywords associated with those roles. Each community groups occupations that share common attributes in tasks, KSAs, and activities, such as administrative and supervisory roles (Community 0) or technical and engineering roles (Community 1). The keywords highlight the primary themes that characterize each community.*

Centrality Analysis

To gain a deeper understanding of the occupational similarity network, I analysed four centrality measures: Degree Centrality, Betweenness Centrality, Closeness Centrality, and Eigenvector Centrality. These measures collectively highlight the most influential and strategically positioned occupations within the network.

Closeness Centrality

$$\text{Formula: } C_C(v) = \frac{N-1}{\sum_{u \neq v} d(v,u)}$$

Closeness Centrality measures how efficiently a node v can reach all other nodes u in a network, where $d(v,u)$, is the shortest path distance between v and u, and N is the total number of nodes (Emmons, et. al 2016).

Occupation	Closeness Centrality
Administrative Services Managers	0,315
First-Line Supervisors of Helpers, Laborers	0,313
First-Line Supervisors of Production and Operating Workers	0,307
Weighers, Measurers, Checkers, and Samplers	0,306
Facilities Managers	0,302
Industrial Production Managers	0,301
First-Line Supervisors of Farming, Fishing, and Forestry Workers	0,300
First-Line Supervisors of Housekeeping and Janitorial Workers	0,299
Production, Planning, and Expediting Clerks	0,297
Food Service Managers	0,297

Table 29: Top 10 Occupations by Closeness Centrality: *This table highlights occupations with the highest closeness centrality scores, indicating roles that efficiently interact with many others in the network. Administrative Services Managers top the list, reflecting their central role in coordinating diverse occupational connections.*

Notably, "Administrative Services Managers" top this measure, followed closely by various first-line supervisory roles and "Weighers, Measurers, Checkers, and Samplers". These occupations are likely facilitating connectivity due to their roles in coordinating and managing multiple departments or teams.

Eigenvector Centrality

$$\text{Formula: } x_i = \frac{1}{\lambda}\sum_{j=1}^{N} A_{ij}x_j$$

Eigenvector Centrality measures a node's influence based on its connections to other influential nodes. In this formula, A_{ij} is the adjacency matrix, λ is the largest eigenvalue, and x_i is the centrality score of node i (Valente, et. al 2008). In the analysis, technical positions such as Welding, Soldering, and Brazing Machine Setters, Operators, and Tenders, Machinists, and Cutting, Punching, and Press Machine Setters, rank highly. These roles are integral to manufacturing and production processes, linking various technical and operational functions within the network and thereby amplifying their influence.

Occupation	Eigenvector Centrality
Welding, Soldering, and Brazing Machine Setters	0,240
Machinists	0,236
Cutting, Punching, and Press Machine Setters	0,234
Grinding and Polishing Workers, Hand	0,197
Multiple Machine Tool Setters, Operators	0,193
Molding, Coremaking, and Casting Machine Setters	0,187
Milling and Planing Machine Setters, Operators	0,173
Structural Metal Fabricators and Fitters	0,154
Grinding, Lapping, Polishing, and Buffing Machine Tool Setters	0,146
Cutting and Slicing Machine Setters, Operators	0,144

Table 30: Top 10 Occupations by Eigenvector Centrality: *This table shows occupations with the highest eigenvector centrality scores, representing roles connected to other influential roles. Technical positions like Welding, Soldering, and Brazing Machine Setters rank highly due to their integral roles in manufacturing and production processes.*

Betweenness Centrality

Formula: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$

Betweenness Centrality quantifies the extent to which a node v lies on the shortest paths between other nodes s and t. Here, σ_{st} is the total number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of those paths that pass-through v (Emmons, et. al 2016). Administrative Services Managers again emerge as top performers, underscoring their role in bridging diverse occupational groups. Additionally, specialized roles like Dental Laboratory Technicians and Librarians and Media Collections Specialists also exhibit significant betweenness centrality, indicating their importance in connecting niche occupational clusters and enabling collaboration across distinct fields.

Occupation	Betweenness Centrality
Administrative Services Managers	0,057
Weighers, Measurers, Checkers, and Samplers	0,039
Helpers--Production Workers	0,029
Dental Laboratory Technicians	0,027
Librarians and Media Collections Specialists	0,025
Insurance Appraisers, Auto Damage	0,025
First-Line Supervisors of Helpers, Laborers	0,025
Inspectors, Testers, Sorters, Samplers, and Weighers	0,024
Residential Advisors	0,023
Museum Technicians and Conservators	0,023

Table 31: Top 10 Occupations by Betweenness Centrality: *This table displays occupations with the highest betweenness centrality scores, identifying key roles that act as bridges between different communities. Administrative Services Managers lead this measure, emphasizing their role in linking diverse occupational groups.*

Appendix

Degree Centrality

Formula: $\boldsymbol{C_D(v) = \deg(v) = \sum_{i=1}^{N} A_{vi}}$

Degree Centrality counts the number of direct connections a node vvv has. In this formula, $\boldsymbol{A_{vi}}$ represents the adjacency matrix, and $de\ g(v)$ is the degree of node v (Emmons, et. al 2016).

In the analysis, Administrative Services Managers lead, followed by technical roles such as Welding, Soldering, and Brazing Machine Setters, Operators, and Tenders and Electrical Engineers. High degree centrality suggests that these occupations are highly connected and versatile, interacting with a broad range of other roles and thus playing a central role in the occupational networks.

Occupation	Degree Centrality
Administrative Services Managers	0,057
Welding, Soldering, and Brazing Machine Setters	0,047
Electrical Engineers	0,046
Machinists	0,043
Grinding and Polishing Workers, Hand	0,041
Cutting, Punching, and Press Machine Setters	0,040
Helpers - Production Workers	0,040
Nurse Practitioners	0,039
Acute Care Nurses	0,038
Loading and Moving Machine Operators, Underground	0,036

Table 32: Top 10 Occupations by Degree Centrality: *This table lists occupations with the highest degree centrality scores, showing roles with numerous direct connections in the network. Administrative Services Managers and various technical roles demonstrate their versatility and broad interaction within the occupational network.*

Appendix G - Building a Skills-First Ecosystem

G.1 Codebase overview

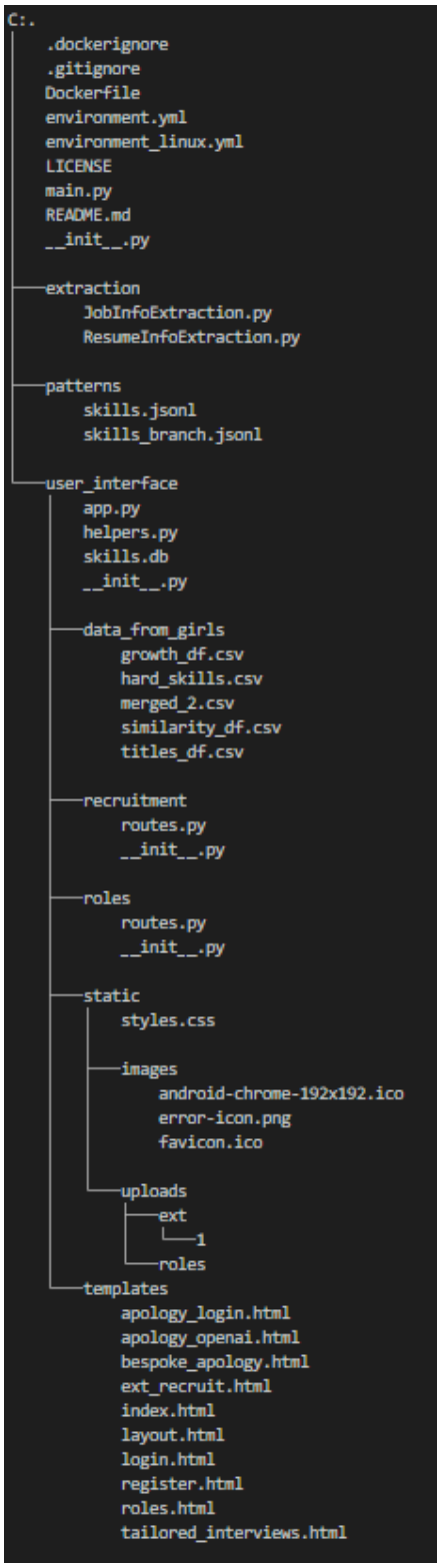


Figure 64: *The file tree shows a comprehensive view of our entire code base. Main.py includes the main logic for all our functionalities. App.py contains our flask application and most of the routes, with the routes for external recruiting and role similarity being located in their respective blueprint directories. Finally, the flask templates are located inside the templates directory and contains our HTML-templates that are used to render our application on the web.*

G.2 Matching model

G.2.1 Web Application

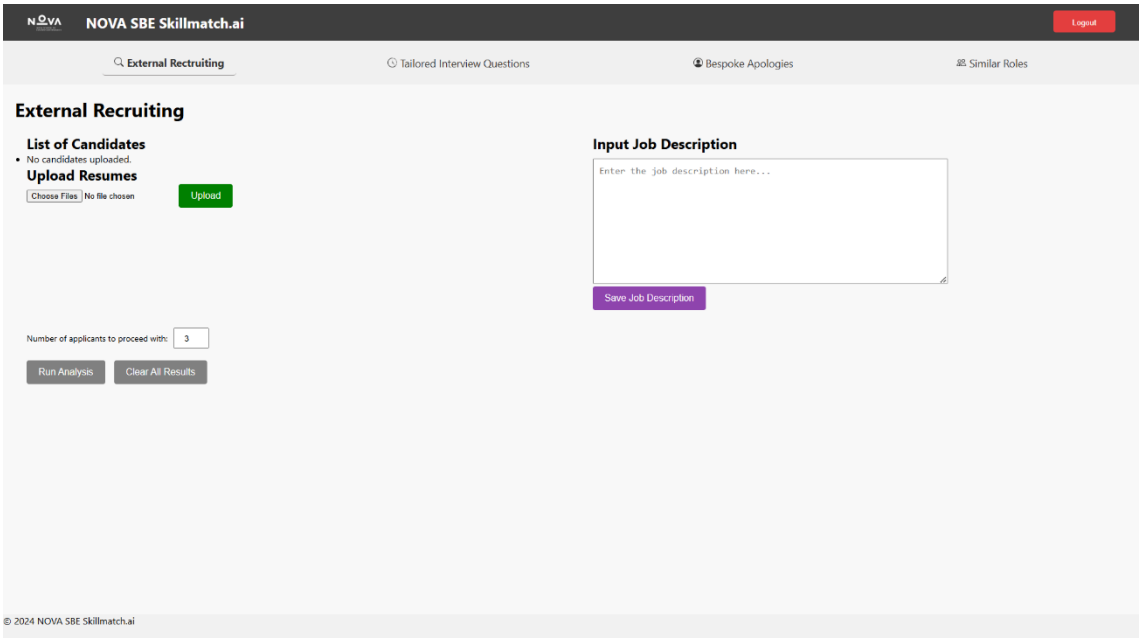


Figure 65: *The external recruiting endpoint before running the analysis. The user has the option to upload several resumes in PDF-format and a job description in the box. After selecting the number of users that should proceed to the interviews, the user will be able to run the analysis.*

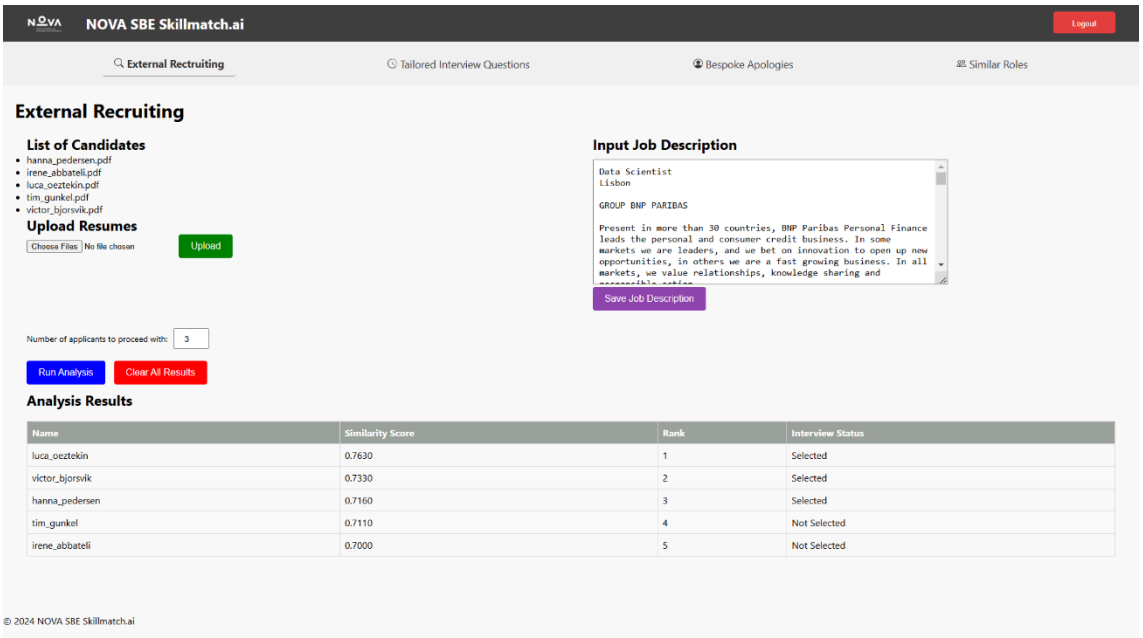


Figure 66: *The external recruitment endpoint after running the analysis. The output is a ranked similarity table which includes the interview status of the applicant. This interview status will be used as a basis for the tailored interview questions and bespoke apology letters.*

G.2.2 main.py

***Code 8**: This function calculates skill similarity between applicants and job postings using BERT-based embeddings (all-MiniLM-L6-v2) and cosine similarity. The algorithm iterates through job descriptions and applicant resumes, generating similarity scores for each pair. Applicants are ranked based on their scores, with the top N candidates classified as "Selected" for further evaluation. This method ensures a fair and skills-driven candidate ranking process.*

```
def calc_similarity(applicant_df, job_df, N=3, parallel=False):
    """Calculate cosine similarity based on BERT embeddings of skills (skill-by-skill)."""

    def semantic_similarity_all_MiniLM_L6_v2(job, resume):
        """Calculate similarity with all-MiniLM-L6-v2."""
        model = SentenceTransformer('all-MiniLM-L6-v2')
        model.eval()
        score = 0
        sen = job + resume
        sen_embeddings = model.encode(sen,
                                      batch_size=4,
                                      device='cpu',
                                      show_progress_bar=False,
                                      num_workers=os.cpu_count() // 2 if parallel else 0
                                      )
        for i in range(len(job)):
            if job[i] in resume:
                score += 1
            else:
                max_cosine_sim = max(cosine_similarity([sen_embeddings[i]], sen_embeddings[len(job):])[0])
                if max_cosine_sim >= 0.4:
                    score += max_cosine_sim
        score = score / len(job)
        return round(score, 3)

    # Prepare a DataFrame to store results
    matching_dataframe = []

    # Loop through each job in the job_df
    for job_index in range(len(job_df)):
        job_skills = job_df['Skills'].iloc[job_index]  # Use iloc for positional indexing

        # Loop through each applicant in the applicant_df
        for applicant_id in range(len(applicant_df)):
            applicant_skills = applicant_df['Skills'].iloc[applicant_id]  # Use iloc for positional indexing
            applicant_name = applicant_df['name'].iloc[applicant_id]  # Ensure correct column access

            # Compute similarity score
            score = semantic_similarity_all_MiniLM_L6_v2(job_skills, applicant_skills)

            # Append result to the DataFrame
            matching_dataframe.append({
                "name": applicant_name,
                "job_id": job_index,
                "similarity_score": score
            })

    # Create a DataFrame from results
    matching_dataframe = pd.DataFrame(matching_dataframe)

    # Add rank based on similarity score
    matching_dataframe['rank'] = matching_dataframe['similarity_score'].rank(ascending=False)
    matching_dataframe['interview_status'] = matching_dataframe['rank'].apply(lambda x: 'Selected' if x <= N else
'Not Selected')

    return matching_dataframe.sort_values("rank")
```

G.2.3 recruitment/routes.py

***Code 9**: These Flask routes handle the external recruitment workflow. The /ext_recruit route serves as the main interface, displaying uploaded resumes and job descriptions. The /upload_files endpoint allows users to upload candidate résumés, while /submit_job_description accepts the job description input. The /run_analysis route performs the skill-matching process by invoking the calc_similarity function, storing results in the database, and rendering ranked candidates. The /clear route resets the session, database, and uploaded files, ensuring a clean state for subsequent analyses. For the external recruitment endpoint we employed a blueprint to keep the codebase modular and scalable.*

```
import os
import sys
import shutil
sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from flask import Blueprint, render_template, request, redirect, url_for, flash, session, current_app
from helpers import allowed_file, get_db, login_required
from werkzeug.utils import secure_filename
import json
import pandas as pd
import main

recruitment_bp = Blueprint('recruitment', __name__, template_folder='../templates')

@recruitment_bp.route('/ext_recruit', methods=['GET'])
@login_required
def ext_recruit():
    user_id = str(session.get("user_id"))
    upload_folder = os.path.join(current_app.config['UPLOAD_FOLDER_EXT'], user_id)
    os.makedirs(upload_folder, exist_ok=True)
    applicants = [file for file in os.listdir(upload_folder) if file.endswith("pdf")]
    job_description = session.get('job_description', '')

    # Initialize analysis_data as empty
    analysis_data = []
    columns = []

    # Connect to the database
    db = get_db()
    cursor = db.cursor()

    # Check if analysis results exist in the database
    cursor.execute("""
        SELECT name, similarity_score, rank, interview_status
        FROM applicants
        WHERE similarity_score IS NOT NULL
        ORDER BY rank ASC
    """)
    ranked_applicants = cursor.fetchall()

    if ranked_applicants:
        # Prepare data for template
        analysis_data = [{
            'name': row[0],
            'similarity_score': round(row[1], 3),
            'rank': row[2],
            'interview_status': row[3]
        } for row in ranked_applicants]
        columns = ['name', 'similarity_score', 'rank', 'interview_status']

    return render_template(
        "ext_recruit.html",
        applicants=applicants,
        analysis_data=analysis_data,
        job_description=job_description,
        columns=columns
    )

@recruitment_bp.route('/ext_recruit/upload_files', methods=['POST'])
@login_required
def upload_files():
    user_id = str(session.get("user_id"))
    user_folder = os.path.join(current_app.config['UPLOAD_FOLDER_EXT'], user_id)
    os.makedirs(user_folder, exist_ok=True)
    files = request.files.getlist('files')
    if not files or files[0].filename == '':
        flash('No files selected')
        return redirect(url_for('recruitment.ext_recruit'))

    for file in files:
        if allowed_file(file.filename):
            filename = secure_filename(file.filename)
            upload_path = os.path.join(current_app.config['UPLOAD_FOLDER_EXT'], user_id, filename)
```

```
            file.save(upload_path)
        else:
            flash(f'File type not allowed: {file.filename}')
            return redirect(url_for('recruitment.ext_recruit'))

    flash('Files successfully uploaded')
    return redirect(url_for('recruitment.ext_recruit'))

@recruitment_bp.route('/ext_recruit/submit_job_description', methods=['POST'])
@login_required
def submit_job_description():
    job_description = request.form.get('job_description', '')
    if not job_description:
        flash('No job description provided')
        return redirect(url_for('recruitment.ext_recruit'))

    session['job_description'] = job_description
    flash('Job description saved')
    return redirect(url_for('recruitment.ext_recruit'))

@recruitment_bp.route('/ext_recruit/run_analysis', methods=['POST'])
@login_required
def run_analysis():
    user_id = str(session.get("user_id"))
    job_description = session.get('job_description')
    upload_folder = os.path.join(current_app.config['UPLOAD_FOLDER_EXT'], user_id)
    applicant_files = [file for file in os.listdir(upload_folder) if file.endswith("pdf")]

    if not applicant_files:
        flash('No applicants to analyze')
        return redirect(url_for('recruitment.ext_recruit'))

    if not job_description:
        flash('No job description provided')
        return redirect(url_for('recruitment.ext_recruit'))

    # Retrieve the number of applicants from the form data
    num_applicants_str = request.form.get('num_applicants', '3')
    try:
        num_applicants = int(num_applicants_str)
        if num_applicants < 1:
            num_applicants = 1
    except ValueError:
        flash('Invalid number of applicants. Using default value of 3.')
        num_applicants = 3

    db = get_db()
    cursor = db.cursor()

    # Rest of your code remains the same, but pass num_applicants to calc_similarity
    # Step 1: Insert Job Posting into the Database
    df_jobs = pd.DataFrame([{'raw': job_description}])
    df_jobs = main.job_info_extraction(df_jobs)

    required_skills = df_jobs.loc[0, 'Skills']
    required_skills_json = json.dumps(required_skills)

    cursor.execute("""
        INSERT INTO job_postings (job_description, required_skills)
        VALUES (?, ?)
    """, (job_description, required_skills_json))
    job_id = cursor.lastrowid

    # Step 2: Insert Applicants into the Database
    df_resumes = main.get_resumes(upload_folder)
    df_resumes = main.resume_extraction(df_resumes)

    for _, row in df_resumes.iterrows():
        name = row.get('name', '')
        contact_info = ''  # Update as needed
        skills = row.get('Skills', [])
        skills_json = json.dumps(skills)
        degrees = row.get('Degrees', [])
        degrees_json = json.dumps(degrees)

        cursor.execute("""
            INSERT OR REPLACE INTO applicants (name, contact_info, skills, degrees)
            VALUES (?, ?, ?, ?)
        """, (name, contact_info, skills_json, degrees_json))

    db.commit()

    # Step 3: Perform Similarity Analysis
```

```
    matching_dataframe = main.calc_similarity(df_resumes, df_jobs, N=num_applicants, parallel=True)

    # Update applicants with similarity scores and ranks
    for _, row in matching_dataframe.iterrows():
        name = row['name']
        similarity_score = row['similarity_score']
        rank = int(row['rank'])
        interview_status = row['interview_status']

        cursor.execute("""
            UPDATE applicants
            SET similarity_score = ?, rank = ?, interview_status = ?
            WHERE name = ?
        """, (similarity_score, rank, interview_status, name))

    db.commit()

    # Step 4: Retrieve Ranked Applicants for Display
    cursor.execute("""
        SELECT name, similarity_score, rank, interview_status
        FROM applicants
        ORDER BY rank ASC
    """)
    ranked_applicants = cursor.fetchall()

    # Prepare data for template
    analysis_data = [{
        'name': row[0],
        'similarity_score': round(row[1], 3),
        'rank': row[2],
        'interview_status': row[3]
    } for row in ranked_applicants]

    columns = ['name', 'similarity_score', 'rank', 'interview_status']

    return render_template(
        "ext_recruit.html",
        applicants=applicant_files,
        analysis_data=analysis_data,
        job_description=job_description,
        columns=columns,
        num_applicants=num_applicants
    )

@recruitment_bp.route('/ext_recruit/clear', methods=['POST'])
@login_required
def clear_results():
    user_id = str(session.get("user_id"))
    upload_folder = os.path.join(current_app.config['UPLOAD_FOLDER_EXT'], user_id)
    db = get_db()
    cursor = db.cursor()
    try:
        # Delete all files in the uploads directory
        for filename in os.listdir(upload_folder):
            file_path = os.path.join(upload_folder, filename)
            if os.path.isfile(file_path):
                os.unlink(file_path)

        # Clear database tables
        cursor.execute("DELETE FROM applicants")
        cursor.execute("DELETE FROM job_postings")
        db.commit()

        # Clear session data
        session.pop('job_description', None)

        flash('All uploaded files and results have been cleared.')
    except Exception as e:
        flash(f'An error occurred while clearing files: {e}')
    return redirect(url_for('recruitment.ext_recruit'))
```

G.2.4 ext_recruit.html

***Code 10**: This Flask template renders the external recruitment interface, enabling users to upload resumes, input job descriptions, and view similarity analysis results. It includes dynamic candidate lists, job description forms, and result tables, with buttons to run or clear analysis. A loading spinner indicates progress during processing. We exclude the style blocks here as they focus on visual presentation rather than core functionality.*

```
{% extends "layout.html" %}

{% block title %}
External Recruiting
{% endblock %}

{% block main %}
<h1>External Recruiting</h1>

<!-- Wrap the main content in a content wrapper -->
<div class="content-wrapper">
    <!-- Candidates and Upload Section (Left) -->
    <div class="flex-container">
        <div class="left-section">
            <div>
                <h2>List of Candidates</h2>
                <ul>
                    {% for applicant in applicants %}
                        <li>{{ applicant }}</li>
                    {% else %}
                        <li>No candidates uploaded.</li>
                    {% endfor %}
                </ul>
            </div>

            <!-- Upload Resumes Button -->
            <div>
                <h2>Upload Resumes</h2>
                <form action="{{ url_for('recruitment.upload_files') }}" method="post" enctype="multipart/form-data">
                    <input type="file" name="files" multiple required>
                    <input type="submit" value="Upload" class="action-button upload-button">
                </form>
            </div>
        </div>

        <!-- Job Description Section (Right) -->
        <div class="right-section">
            <h2>Input Job Description</h2>
            <form method="post" action="{{ url_for('recruitment.submit_job_description') }}">
                <textarea name="job_description" rows="10" cols="50" placeholder="Enter the job description here..."
required>{{ job_description or '' }}</textarea>
                <br>
                <input type="submit" value="Save Job Description" class="action-button job-desc-button">
            </form>
        </div>
    </div>

   <!-- Buttons Section (Run Analysis + Clear Results) -->
   <form method="post" action="/ext_recruit/run_analysis" id="analysis-form" class="analysis-form">
    <div class="input-group">
        <label for="num_applicants">Number of applicants to proceed with:</label>
        <input type="number" id="num_applicants" name="num_applicants" min="1" value="3" required>
    </div>
    <div class="buttons-container">
        <!-- Input Field and Run Analysis Button -->

            <button type="submit" class="action-button run-button" {% if not applicants or not job_description
%}disabled{% endif %}>
                Run Analysis
            </button>
        </form>

        <!-- Clear All Results Button -->
        <form method="post" action="/ext_recruit/clear" onsubmit="return confirm('Are you sure you want to delete all
uploaded resumes and clear results?');">
            <button type="submit" class="action-button clear-button" {% if not applicants %}disabled{% endif %}>
                Clear All Results
            </button>
        </form>
    </div>

    <!-- Analysis Results -->
    <div class="analysis-results">
        {% if analysis_data %}
            <h2>Analysis Results</h2>
```

```
            <table class="table table-striped">
                <thead>
                    <tr>
                        {% for column in columns %}
                            <th>{{ column.replace('_', ' ').title() }}</th>
                        {% endfor %}
                    </tr>
                </thead>
                <tbody>
                    {% for row in analysis_data %}
                        <tr>
                            {% for column in columns %}
                                <td>
                                    {% if column == "similarity_score" %}
                                        {{ "{:.4f}".format(row[column]) }}
                                    {% else %}
                                        {{ row[column] }}
                                    {% endif %}
                                </td>
                            {% endfor %}
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        {% endif %}
    </div>
</div> <!-- End of content-wrapper -->

<!-- Loading Spinner Overlay -->
<div id="loading-overlay">
    <div class="spinner"></div>
    <p>Running analysis, please wait...</p>
</div>

{% endblock %}


{% block scripts %}
<script>
    // Show the loading overlay when the analysis form is submitted
    document.getElementById('analysis-form').addEventListener('submit', function() {
        document.getElementById('loading-overlay').style.display = 'block';
    });
</script>
{% endblock %}
```

G.3 Tailored Interview Questions

G.3.1 Web Application

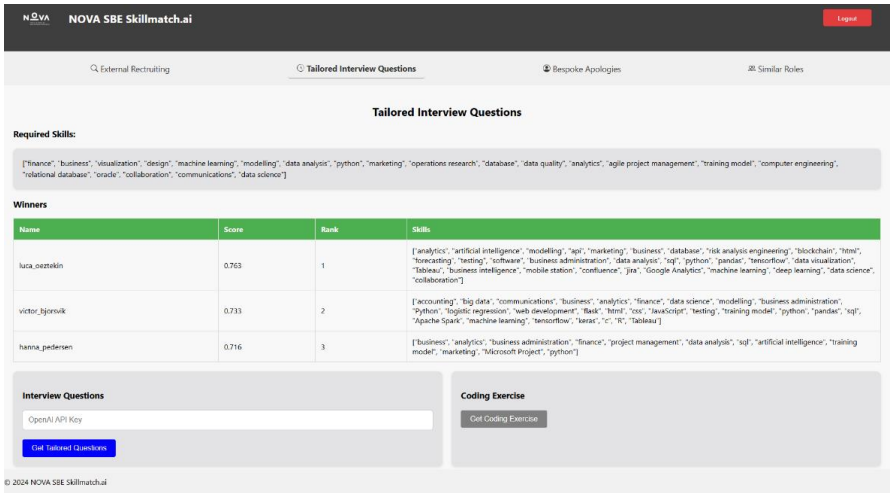


Figure 67: *The tailored interview questions endpoint before generating the questions and exercise. The Required Skills list highlights all the required skills from the job description while the table shows the shortlisted applicants and the skills they possess. If the active user is the admin user, there is no need to input the OpenAI API key. Otherwise, the user should input their API key before generating the questions and exercise.*

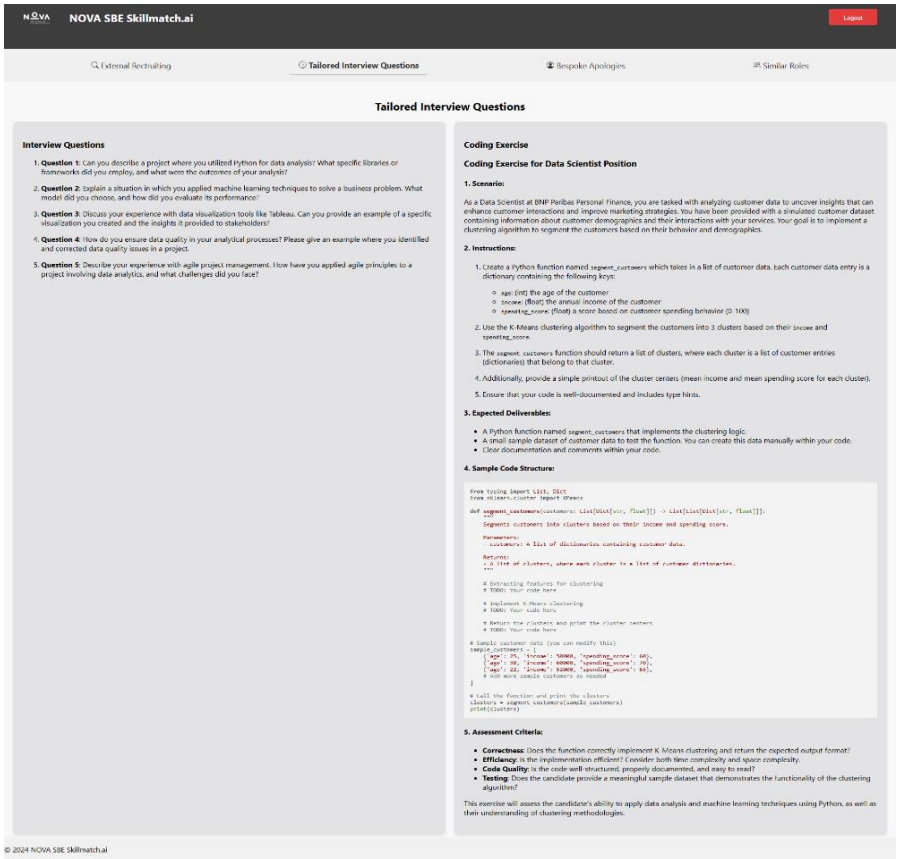


Figure 68: *The tailored interview questions after generating the questions and exercise. The results are displayed in markdown and the code is provided in the relevant syntax and highlighting for the coding language thanks to the highlight.js and marked.js library.*

G.3.2 main.py

***Code 11**: These two functions generate tailored interview questions and coding exercises based on candidate skills and job requirements. Using OpenAI's GPT API, the tailored_questions function creates a standardized list of five self-contained interview questions, ensuring fairness and alignment with the job's responsibilities. Similarly, generate_coding_exercise produces a coding task tailored to the job description, complete with instructions, deliverables, and assessment criteria. Both functions ensure dynamic, skill-based evaluations to support objective recruitment processes.*

```
@retry(wait=wait_random_exponential(min=1, max=60), stop=stop_after_attempt(6))
def tailored_questions(api_key, applicants, required_skills, model="gpt-4o-mini"):
    """
    Function to create tailored interview questions using OpenAI's API.

    :param api_key: str, the OpenAI API key for authentication.
    :param applicants: str, a list of candidates' names and their skills.
    :param required_skills: str, the skills required for the job.
    :param model: str, the OpenAI model to use (default: "gpt-4o-mini").
    :return: str, HTML content of the tailored interview questions.
    """
    prompt = f"""
    We have a list of candidates and wish to generate tailored interview questions based on their skills and the
skills required for the job. Note that all candidates should receive the same set of questions to ensure fairness and
avoid bias.

    Please generate a list of **5 self-contained interview questions** that:
    - Assess the overlap between the candidates' skills and the required job skills.
    - Are relevant to the job's responsibilities and the provided skillsets.
    - Cover both technical skills and problem-solving abilities, where applicable.
    - Are phrased clearly and concisely.

    Here is the information:
    - **Candidates and their skills**: {applicants}
    - **Required job skills**: {required_skills}

    The output should follow this structure:
    1. **Question 1**: [Write the first question.]
    2. **Question 2**: [Write the second question.]
    3. **Question 3**: [Write the third question.]
    4. **Question 4**: [Write the fourth question.]
    5. **Question 5**: [Write the fifth question.]

    Each question should challenge the candidates to demonstrate their proficiency in the relevant areas while being
answerable based on their stated skills.
    """

    # Call OpenAI's API to generate the response
    client = OpenAI(api_key=api_key)
    completion = client.chat.completions.create(
        model=model,
        messages=[
            {"role": "system", "content": "You are a helpful recruiting assistant. Your job is to generate fair,
skill-based interview questions tailored to a set of candidates' skills and the job's requirements."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7,
        max_tokens=1000
    )
    markdown_output = completion.choices[0].message.content
    html_output = markdown.markdown(markdown_output)  # Convert markdown to HTML

    return html_output

@retry(wait=wait_random_exponential(min=1, max=60), stop=stop_after_attempt(6))
def generate_coding_exercise(api_key, job_description, model="gpt-4"):
    """
    Generate a tailored coding exercise based on a job description using OpenAI's API.

    :param job_description: str, the job description to base the exercise on
    :return: str, the coding exercise prompt
    """
    prompt = f"""
    Based on the following job description, create a self-contained practical coding exercise for a coding interview.
The exercise should:
    - Be solvable using Python.
    - Be tailored to the job description's requirements and responsibilities.
    - Include clear instructions for the candidate.
    - Require no external dataset and be solvable based on the prompt alone.
    - Follow this structure:

    1. **Scenario**: Provide a realistic scenario relevant to the job.
    2. **Instructions**: Step-by-step guidance on what the candidate needs to do.
    3. **Expected Deliverables**: Clearly define what is expected from the candidate.
```

```
    4. **Sample Code Structure**: Offer a template or skeleton code for the candidate to follow.
    5. **Assessment Criteria**: Outline how the solution will be evaluated.

    Job Description:
    {job_description}

    Generate the exercise now.
    """

    # Call OpenAI's API to generate the response
    client = OpenAI(api_key=api_key)
    completion = client.chat.completions.create(
        model=model,
        messages=[
            {"role": "system", "content": "You are a helpful assistant that generates coding exercises for job
interviews."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7,
        max_tokens=1000
    )

    markdown_output = completion.choices[0].message.content
    return markdown_output
```

G.3.3 app.py

***Code 12**: These Flask routes handle generating tailored interview questions and coding exercises using the OpenAI API. The /tailored_interviews route creates skill-relevant questions for selected applicants, while /tailored_interviews/coding_exercise produces coding tasks based on job descriptions.*

```
@app.route("/tailored_interviews", methods=["GET", "POST"])
@login_required
def tailored_interviews():
    """Display tailored interviews for selected applicants."""
    # Fetch data
    db = get_db()
    cursor = db.cursor()
    cursor.execute(
        """SELECT * FROM applicants
          WHERE interview_status = 'Selected'
          ORDER BY RANK""")
    winners = cursor.fetchall()
    cursor.execute("SELECT required_skills FROM job_postings")
    required_skills = cursor.fetchall()

    if request.method == 'POST':
        if winners:
            MODEL = "gpt-4o-mini"

            if session.get("user_id") == 1:
                api_key = os.getenv("OPENAI_API_KEY", "<your OpenAI API key if not set as an env var>")
            else:
                api_key = request.form.get("password")
            try:
                # Retrieve tailored interview questions
                response = main.tailored_questions(api_key, winners, required_skills, model=MODEL)
            except tenacity.RetryError as e:
                return apology_openai("Need a valid OpenAI API Key", 403)

        else:
            response = []
        winners = []
        required_skills = []
        return render_template("tailored_interviews.html", winners=winners, response=response,
required_skills=required_skills)
    else:
        return render_template("tailored_interviews.html", winners=winners,
required_skills=required_skills)

@app.route("/tailored_interviews/coding_exercise", methods=["POST"])
@login_required
def coding_exercise():
    """Generate coding exercise based on job description."""
```

```
    # Fetch data
    db = get_db()
    cursor = db.cursor()
    cursor.execute("SELECT job_description FROM job_postings")
    job_description = cursor.fetchone()
    MODEL = "gpt-4o-mini"

    if session.get("user_id") == 1:
        api_key = os.getenv("OPENAI_API_KEY", "<your OpenAI API key if not set as an env var>")
    else:
        api_key = request.form.get("password")
    try:
        # Retrieve tailored interview questions
        coding_exercise_output= main.generate_coding_exercise(api_key, job_description, model=MODEL)
    except tenacity.RetryError as e:
        return jsonify({"error": "Invalid API Key"}), 403

    return jsonify({"coding_exercise": coding_exercise_output})
```

3.4 tailored_interviews.html

***Code 13**: This HTML template renders the tailored interview questions and coding exercise functionalities. The interview questions are server-rendered based on the API response, while the coding exercise is dynamically fetched and displayed using JavaScript. Block styles were excluded here as they were not directly relevant to the template's functionality.*

```
{% extends "layout.html" %}

{% block title %}
Tailored Questions
{% endblock %}

{% block main %}
<p class="hello-user">Tailored Interview Questions</p>

<!-- If winners exist, show required skills at the top -->
{% if winners and required_skills %}
<h3>Required Skills:</h3>
<div class="skills-box">
    {{ required_skills[0][0] }}
</div>
{% endif %}

<!-- If winners exist, show the table next -->
{% if winners %}
<div class="winners-table-container">
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Score</th>
                <th>Rank</th>
                <th>Skills</th>
            </tr>
        </thead>
        <tbody>
        {% for winner in winners %}
        <tr>
            <td>{{ winner[1] }}</td>
            <td>{{ winner[5] | round(5, 'floor') }}</td>
            <td>{{ winner[6] }}</td>
            <td>{{ winner[3] }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
{% endif %}

<!-- Loading Spinner Overlay -->
<div id="loading-overlay">
    <div class="spinner"></div>
    <p>Making Tailored Interview Questions, please wait...</p>
</div>

<!-- Side-by-side containers for Interview Questions and Coding Exercise -->
<div class="content-row">
    <div class="content-box">
        <h3>Interview Questions</h3>
```

```
        {% if response %}
            <div class="questions-content">
                {{ response | safe }}
            </div>
        {% endif %}

        <!-- Form for getting tailored questions -->
        <form method="post" action="{{ url_for('tailored_interviews') }}" id="analysis-form" class="analysis-form">
            {% if not response %}
            <div class="form-group">
                <label for="password"></label>
                <input id="password" name="password" placeholder="OpenAI API Key" type="password">
            </div>
            <input type="hidden" name="action" value="run_analysis">
            <button type="submit" class="action-button run-button" {% if not winners or not required_skills
%}disabled{% endif %}>Get Tailored Questions</button>
            {% endif %}
        </form>

    </div>

    <div class="content-box">
        <h3>Coding Exercise</h3>
        <div id="coding-exercise-container">
            <!-- Initially empty, will be filled by JS after fetch -->
        </div>
        <button id="fetch-exercise-btn" class="action-button run-button"
            {% if not response %}disabled{% endif %}>Get Coding Exercise</button>
    </div>
    </div>
</div>

{% endblock %}

{% block scripts %}
<script>
    // Show the loading overlay when the analysis form is submitted
    const analysisForm = document.getElementById('analysis-form');
    if (analysisForm) {
        analysisForm.addEventListener('submit', function() {
            document.getElementById('loading-overlay').style.display = 'block';
        });
    }

    // Fetch Coding Exercise
    const fetchExerciseBtn = document.getElementById('fetch-exercise-btn');
    if (fetchExerciseBtn) {
        fetchExerciseBtn.addEventListener('click', function() {
            const container = document.getElementById('coding-exercise-container');
            container.innerHTML = "<p>Loading coding exercise...</p>";

            const formData = new FormData();
            const passwordInput = document.getElementById('password');
            if (passwordInput) {
                formData.append('password', passwordInput.value);
            }

            fetch("{{ url_for('coding_exercise') }}", {
                method: "POST",
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                if (data.error) {
                    container.innerHTML = `<p>Error: ${data.error}</p>`;
                } else {
                    // Use marked.parse() to convert Markdown
                    const html = marked.parse(data.coding_exercise);
                    container.innerHTML = html;
                    hljs.highlightAll();
                    fetchExerciseBtn.remove();
                }
            })
            .catch(error => {
                container.innerHTML = `<p>An error occurred: ${error}</p>`;
            });
        });
    }
</script>
{% endblock %}
```

G.4 Bespoke Apologies

G.4.1 Web Application

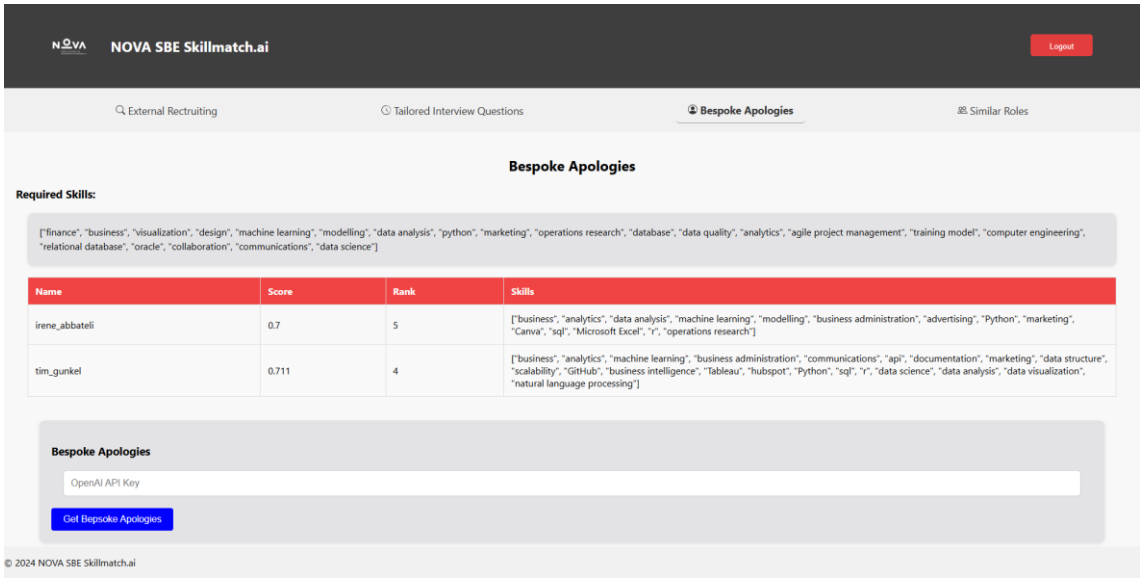


Figure 69: *The bespoke apologies endpoint before generating the bespoke apologies. The Required Skills list highlights all the required skills from the job description while the table shows the applicants that did not meet the requirements and the skills they possess. If the active user is the admin user, there is no need to input the OpenAI API key. Otherwise, the user should input their API key before generating the bespoke apologies.*

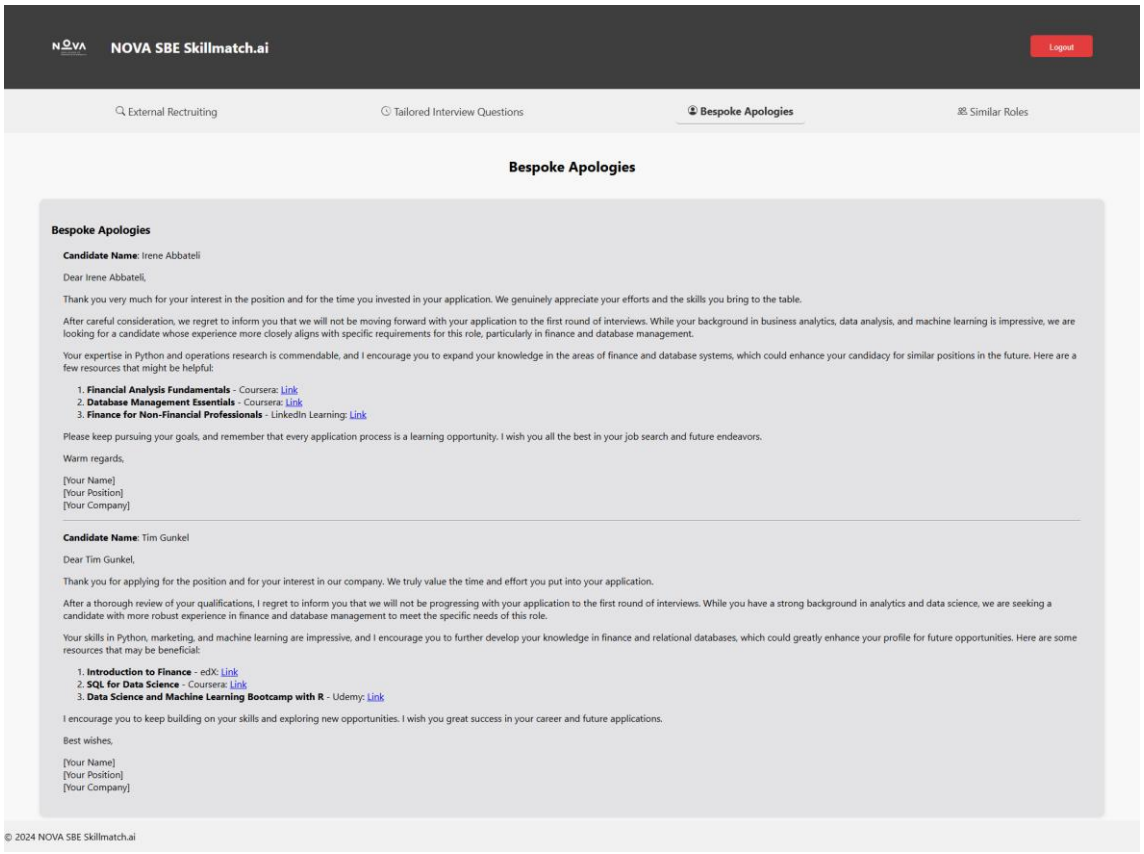


Figure 70: *The bespoke apologies endpoint after generating the apologies. The apologies are rendered in Markdown and provides links to courses where the applicants can improve on the skills they were lacking for the relevant role.*

4.2 main.py

***Code 14**: This function generates personalized apology letters for candidates who were not shortlisted. Using OpenAI's GPT model, it creates empathetic and constructive feedback tailored to each applicant's skills and the job requirements. The letters also provide resources for skill improvement and encouragement for future opportunities.*

```python
@retry(wait=wait_random_exponential(min=1, max=60), stop=stop_after_attempt(6))
def bespoke_apologies(api_key, applicants, required_skills, model="gpt-4o-mini"):
    """
    Function to create bespoke apology letters using OpenAI's API.

    :param api_key: str, the OpenAI API key for authentication.
    :param applicants: str, a list of candidates' names and their skills.
    :param required_skills: str, the skills required for the job.
    :param model: str, the OpenAI model to use (default: "gpt-4o-mini").
    :return: str, HTML content of the bespoke apology letters.
    """
    prompt = f"""
    We have a list of candidates for a job, but unfortunately, none of them made it to the first round of interviews.
    Please generate a **bespoke apology letter** for each candidate that:

    - Addresses the candidate by name.
    - Explains respectfully that their skills were not a perfect match for the job's requirements.
    - Offers encouragement and constructive feedback.
    - Provides specific resources or advice to help them improve their skills for future opportunities.

    Here is the information:
    - **Candidates and their skills**: {applicants}
    - **Required job skills**: {required_skills}

    Each apology letter should:
    - Start with a polite and empathetic opening.
    - Acknowledge the candidate's effort in applying for the role.
    - Highlight their strengths, if applicable.
    - Suggest practical resources (e.g., courses, books, online materials) tailored to the skills they are lacking.
      Provide links to relevant online courses and list them like 1., 2. and 3. on seperate lines.
    - End on an encouraging note, wishing them success in their career.

    Output the letters in a structured format:
    - **Candidate Name**: [Write the bespoke apology letter here.]

    Generate the bespoke apologies now.
    """

    # Call OpenAI's API to generate the response
    client = OpenAI(api_key=api_key)
    completion = client.chat.completions.create(
        model=model,
        messages=[
            {
                "role": "system",
                "content": (
                    "You are a helpful and empathetic recruiting assistant. "
                    "Your task is to create polite, constructive, and encouraging apology letters "
                    "for candidates who did not proceed to the first round of interviews."
                )
            },
            {
                "role": "user",
                "content": prompt
            }
        ],
        temperature=0.7,
        max_tokens=1000
    )

    markdown_output = completion.choices[0].message.content
    html_output = markdown.markdown(markdown_output)  # Convert markdown to HTML

    return html_output
```

4.3 app.py

***Code 15**: This route handles the creation and display of bespoke apology letters for applicants not selected for interviews. When a POST request is made, the route fetches the list of non-selected candidates and required job skills, then uses the bespoke_apologies function to generate personalized feedback. The results, including the apology letters, are rendered on the template.*

```
@app.route("/bespoke_apology", methods=["GET", "POST"])
@login_required
def bespoke_apology():
    """Display bespoke apologies for applicants not selected for interviews."""
    # Fetch data
    db = get_db()
    cursor = db.cursor()
    cursor.execute("SELECT * FROM applicants WHERE interview_status = 'Not Selected'")
    losers = cursor.fetchall()
    cursor.execute("SELECT required_skills FROM job_postings")
    required_skills = cursor.fetchall()

    if request.method == 'POST':
        if losers:
            MODEL = "gpt-4o-mini"

            if session.get("user_id") == 1:
                api_key = os.environ.get("OPENAI_API_KEY")
            else:
                api_key = request.form.get("password")
            try:
                response = main.bespoke_apologies(api_key, losers, required_skills, model=MODEL)
            except tenacity.RetryError as e:
                return apology_openai("Need a valid OpenAI API Key", 403)

        else:
            response = []
        losers = []
        required_skills = []
        return render_template("bespoke_apology.html", losers=losers, response=response,
required_skills=required_skills)
    else:
        return render_template("bespoke_apology.html", losers=losers, required_skills=required_skills)
```

4.4 bespoke_apology.html

***Code 16:** Template for displaying bespoke apologies to non-selected applicants. Required skills and applicant details are server-rendered, while bespoke apologies are generated dynamically upon user input.*

```
{% extends "layout.html" %}

{% block title %}
Bespoke Apologies
{% endblock %}

{% block main %}
<p class="hello-user">Bespoke Apologies</p>

<!-- If losers exist, show required skills at the top -->
{% if losers and required_skills %}
<h3>Required Skills:</h3>
<div class="skills-box">
    {{ required_skills[0][0] }}
</div>
{% endif %}

<!-- If losers exist, show the table next -->
{% if losers %}
<div class="losers-table-container">
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Score</th>
                <th>Rank</th>
                <th>Skills</th>
            </tr>
        </thead>
        <tbody>
        {% for loser in losers %}
        <tr>
            <td>{{ loser[1] }}</td>
```

```
            <td>{{ loser[5] | round(5, 'floor') }}</td>
            <td>{{ loser[6] }}</td>
            <td>{{ loser[3] }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
{% endif %}

<!-- Loading Spinner Overlay -->
<div id="loading-overlay">
    <div class="spinner"></div>
    <p>Making bespoke apologies, please wait...</p>
</div>

<!-- Container for Bespoke Apologies -->
<div class="content-row">
    <div class="content-box">
        <h3>Bespoke Apologies</h3>
        {% if response %}
            <div class="questions-content">
                {{ response | safe }}
            </div>
        {% endif %}

        <!-- Form for getting Bespoke Apologies -->
        <form method="post" action="{{ url_for('bespoke_apology') }}" id="analysis-form" class="analysis-form">
            {% if not response %}
            <div class="form-group">
                <label for="password"></label>
                <input id="password" name="password" placeholder="OpenAI API Key" type="password">
            </div>
            <input type="hidden" name="action" value="run_analysis">
            <button type="submit" class="action-button run-button" {% if not losers or not required_skills
%}disabled{% endif %}>Get Bepsoke Apologies</button>
            {% endif %}
        </form>
    </div>
</div>



{% endblock %}

{% block scripts %}
<script>
    // Show the loading overlay when the analysis form is submitted
    document.getElementById('analysis-form').addEventListener('submit', function() {
        document.getElementById('loading-overlay').style.display = 'block';
    });
</script>
{% endblock %}
```

G.5 Similar roles

G.5.1 Web Application

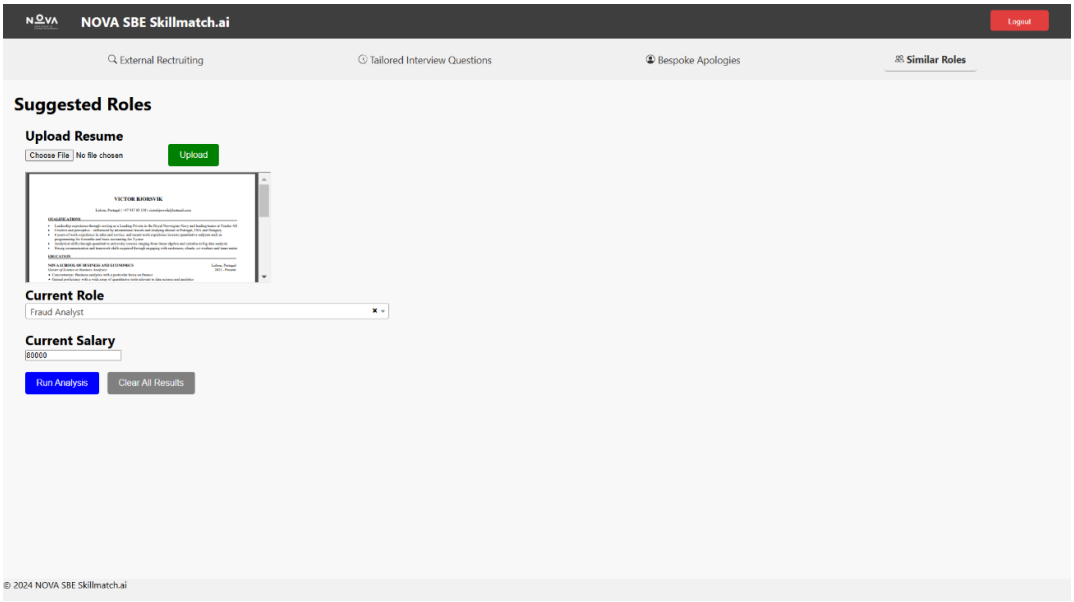


Figure 71: *The Similar Roles endpoint before running the analysis. The user has the option to upload a single résumé in PDF format. After inputting their role from the dropdown menu (data from the O*NET database) and their current salary in USD, the user may run the analysis.*

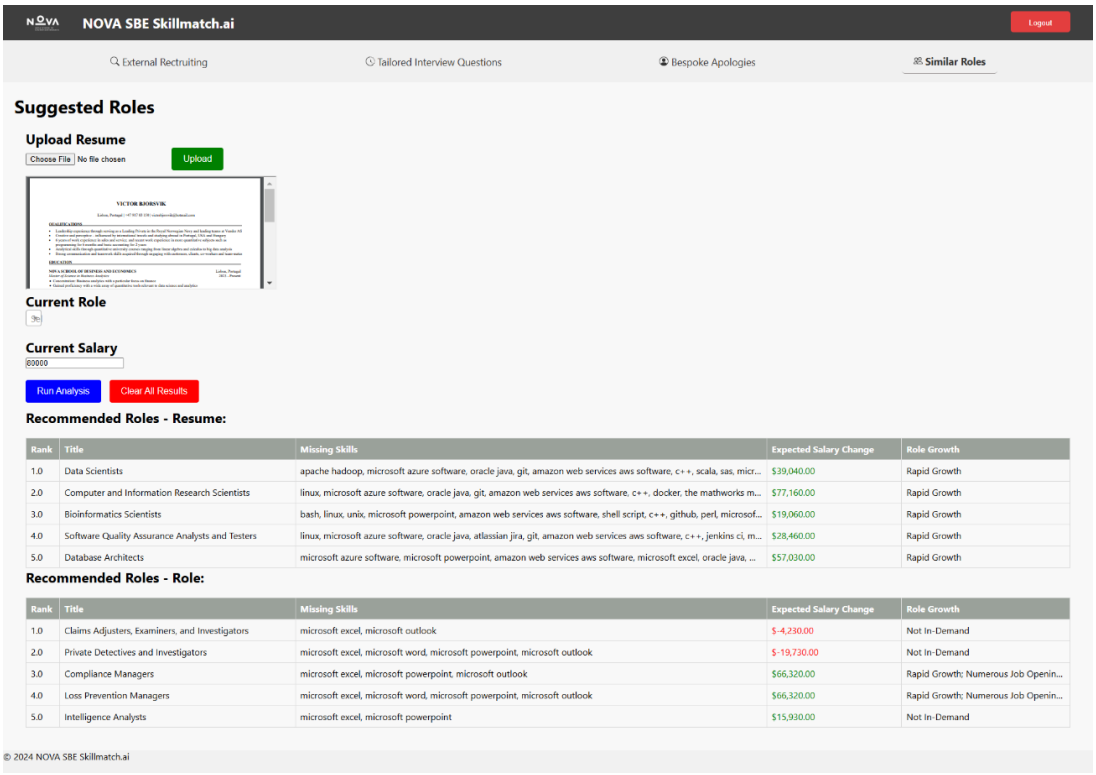


Figure 72: *The Similar Roles endpoint after running the analysis. The output is two tables, one that suggest roles based on the skills extracted from the résumé and another that is based on the input role. The tables highlight the missing skills the applicant should focus on if they want to make the career shift. It also shows the expected salary change in USD and the projected role growth for the role.*

G.5.2 main.py

***Code 17**: Developed in Chapter 8, this function performs role similarity analysis by leveraging BERT-based embeddings and cosine similarity to match a candidate's CV with job descriptions. It calculates normalized similarity scores, identifies missing skills, and integrates role-specific growth and salary variations, providing ranked recommendations.*

```
def role_similarity(cv_df, job_df, hard_skills, scores_df, titles_df, growth_df, role=None, wage=0 , parallel=False):
    """
    Calculate cosine similarity between a single CV and multiple job descriptions.
    Optionally filter based on a specific role and include role-specific scores in the output.
    """
    # Initialize the model
    model = SentenceTransformer('all-mpnet-base-v2')
    model.max_seq_length = 75
    model.tokenizer.padding_side = "right"
    model.eval()

    def add_eos(input_examples):
        """Helper function to add special tokens between each skill."""
        input_examples = [input_example + model.tokenizer.eos_token for input_example in input_examples]
        return input_examples

    def missing_skills(skills, cv):
        """Return a list of words from the job description (title and skills) that do not match any keywords in the
CV."""
        skill_words = set(skill.lower().strip() for skill in skills if isinstance(skill, str))
        cv_set = set(cv.lower().strip() for cv in cv if isinstance(cv, str))
        missing_words = skill_words - cv_set
        return list(missing_words)

    # Find the generic title (Role) corresponding to the input title (Reported Job Title)
    generic_title = titles_df.loc[titles_df['Reported Job Title'] == role, 'Title'].iloc[0]

    # Filter scores_df using the generic title
    filtered_scores_df = scores_df[scores_df['Target_Role'] == generic_title]
    # Map the filtered scores to a dictionary for quick lookup
    role_score_mapping = dict(zip(filtered_scores_df['Role'], filtered_scores_df['composite_tasks_dwas_ksas']))

    # Extract and process the CV's skills
    cv_df['Skills_Text'] = cv_df['Skills'].apply(add_eos)
    cv_df['Skills_Text'] = cv_df['Skills_Text'].apply(
        lambda x: ' '.join(sorted(set(x))) if isinstance(x, list) else ''
    )
    cv_embedding = model.encode(
        cv_df['Skills_Text'].iloc[0],  # Assuming a single CV is provided
        batch_size=1,
        show_progress_bar=False
    )

    # Get job embeddings from DataFrame
    job_embeddings = np.vstack(job_df['skills_embed'].values)

    # Compute cosine similarity
    similarity_scores = cosine_similarity([cv_embedding], job_embeddings).flatten()
    job_df['similarity_score'] = similarity_scores

    # Normalize similarity scores using z-score
    mean_score = job_df['similarity_score'].mean()
    std_score = job_df['similarity_score'].std()
    if std_score > 0:
        job_df['normalized_similarity_score'] = (job_df['similarity_score'] - mean_score) / std_score
    else:
        job_df['normalized_similarity_score'] = 0  # Assign 0 if std is zero (all scores are identical)

    # Rank the jobs by normalized similarity score
    job_df['rank'] = job_df['normalized_similarity_score'].rank(ascending=False)

    # Add missing skills
    job_df['missing_skills'] = job_df.apply(
        lambda row: missing_skills(
            hard_skills.loc[row.name, 'Skills'],  # Skills from df_skill_role_grouped
            cv_df['Skills'].iloc[0]  # Skills from the first row of df_resumes
        ),
        axis=1
    )

    # Map the growth information using generic title
    role_growth_mapping = dict(zip(growth_df['Occupation'], growth_df['Categories']))
    job_df['role_growth'] = job_df['Title'].map(role_growth_mapping)

    # Fill NaN values in 'role_growth' with 'Not In-Demand'
    job_df['role_growth'].fillna('Not In-Demand', inplace=True)
```

```
    # Check for missing salary information
    job_df['annual_wage_variation'] = job_df.apply(
        lambda row: "Info Not Available" if row['annual_wage'] == 0 else row['annual_wage'] - wage,
        axis=1
    )

    job_df_2 = job_df.copy()
    # Add the role-specific scores to a new column
    job_df_2['role_scores'] = job_df_2['Title'].apply(lambda x: role_score_mapping.get(x, None))
    # Rank the jobs by normalized similarity score
    job_df_2['rank'] = job_df_2['role_scores'].rank(ascending=False)

    # Return a DataFrame with the job titles, similarity scores, normalized scores, role-specific scores, and ranks
    ranked_jobs_2 = job_df_2[['rank', 'Title', 'missing_skills', 'annual_wage_variation',
'role_growth']].sort_values(by='rank', ascending=True)
    ranked_jobs = job_df[['rank', 'Title', 'missing_skills', 'annual_wage_variation',
'role_growth']].sort_values(by='rank', ascending=True)

    return ranked_jobs, ranked_jobs_2
```

G.5.3 roles/routes.py

***Code 18**: This code uses Flask blueprints to implement the "Similar Roles" functionality. It includes routes for uploading resumes, running similarity analysis, and clearing data. The run_analysis route integrates the role similarity function from Chapter 8, leveraging pre-loaded DataFrames to compute personalized job recommendations and display them in the template.*

```
import os
import sys
sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from flask import Blueprint, render_template, request, redirect, url_for, flash, session, current_app
from helpers import allowed_file, get_db, login_required
from werkzeug.utils import secure_filename
import pandas as pd
import main

roles_bp = Blueprint('roles', __name__, template_folder='../templates')

@roles_bp.route('/roles', methods=['GET'])
@login_required
def roles():
    user_id = session.get('user_id')
    upload_folder = current_app.config['UPLOAD_FOLDER_ROLES']
    filename = f"user_{user_id}_resume.pdf"
    upload_path = os.path.join(upload_folder, filename)

    if os.path.exists(upload_path):
        applicant = filename  # This will be used to display or link to the resume
    else:
        applicant = None  # No resume uploaded yet

    db = get_db()
    cursor = db.cursor()
    cursor.execute("""
        SELECT role
        FROM unique_roles
        ORDER BY role ASC
    """)
    roles = [role[0] for role in cursor.fetchall()]

    return render_template(
        "roles.html",
        applicant=applicant,
        roles=roles
    )

@roles_bp.route('/roles/upload_files', methods=['POST'])
@login_required
def upload_files():
    file = request.files.get('file')
    if not file or file.filename == '':
        flash('No file selected')
        return redirect(url_for('roles.roles'))

    if allowed_file(file.filename):
        filename = secure_filename(file.filename)
        user_id = session.get('user_id')
        # Ensure user_id is available
        if not user_id:
            flash('User not identified')
            return redirect(url_for('roles.roles'))

        # Save the file with a unique name per user
```

```
        filename = f"user_{user_id}_resume.pdf"
        upload_path = os.path.join(current_app.config['UPLOAD_FOLDER_ROLES'], filename)

        # Optionally, remove the old resume if it exists
        if os.path.exists(upload_path):
            os.remove(upload_path)

        file.save(upload_path)
        flash('Resume successfully uploaded')
    else:
        flash(f'File type not allowed: {file.filename}')
    return redirect(url_for('roles.roles'))

@roles_bp.route('/roles/run_analysis', methods=['POST'])
@login_required
def run_analysis():
    # Get pre-loaded DataFrames
    growth_df = current_app.config['GROWTH_DF']
    df_skill_role_grouped = current_app.config['DF_SKILL_ROLE_GROUPED']
    similarity_df = current_app.config['SIMILARITY_DF']
    titles_df = current_app.config['TITLES_DF']
    merged_df = current_app.config['MERGED_DF']

    # Now you can proceed directly with analysis
    role = request.form.get('current_role')
    wage = float(request.form.get('current_salary'))

    df_resumes = main.get_resumes(current_app.config['UPLOAD_FOLDER_ROLES'])
    df_resumes = main.resume_extraction(df_resumes)
    resume_based, role_based = main.role_similarity(df_resumes, merged_df, df_skill_role_grouped, similarity_df,
titles_df, growth_df, role=role, wage=wage)

    resume_based_col = resume_based.columns
    role_based_col = role_based.columns
    resume_based = resume_based.head(5).to_dict(orient='records')
    role_based = role_based.head(5).to_dict(orient='records')

    return render_template(
        "roles.html",
        applicant=f"user_{session.get('user_id')}_resume.pdf",
        resume_based=resume_based,
        resume_based_col=resume_based_col,
        role_based=role_based,
        role_based_col=role_based_col,
        analysis_done=True
    )

@roles_bp.route('/roles/clear', methods=['POST'])
@login_required
def clear_results():
    # Get pre-loaded DataFrames
    upload_folder = current_app.config['UPLOAD_FOLDER_ROLES']
    try:
        # Delete all files in the uploads directory
        for filename in os.listdir(upload_folder):
            file_path = os.path.join(upload_folder, filename)
            if os.path.isfile(file_path):
                os.unlink(file_path)
        flash('All uploaded files and results have been cleared.')
    except Exception as e:
        flash(f'An error occurred while clearing files: {e}')

    return redirect(url_for('roles.roles'))
```

G.5.4 roles.html

***Code 19**: This template for "Similar Roles" enables users upload resumes, select current roles and salaries, and view personalized role recommendations. Results are displayed in two tables: one based on resumes and another on current roles. JavaScript libraries including marked.js and highlight.js handles searchable dropdowns and loading overlays. Block styles are excluded for conciseness.*

```
{% extends "layout.html" %}

{% block title %}
Suggested Roles
{% endblock %}

{% block main %}
<h1>Suggested Roles</h1>

<!-- Wrap the main content in a content wrapper -->
<div class="content-wrapper">
    <!-- Candidates and Upload Section (Left) -->
    <div class="flex-container">
        <div class="left-section">
            <!-- Upload Resumes Button -->
            <div>
                <h2>Upload Resume</h2>
                <form action="{{ url_for('roles.upload_files') }}" method="post" enctype="multipart/form-data">
                    <input type="file" name="file" required>
                    <input type="submit" value="Upload" class="action-button upload-button">
                </form>
                    {% if applicant %}
                    <!-- Embed the PDF directly -->
                    <iframe
                        src="{{ url_for('static', filename='uploads/roles/' + applicant) }}"
                        width="50%"
                        height="200px"
                    >
                    </iframe>
                {% else %}
                    <p>No resume uploaded yet.</p>
                {% endif %}
            </div>
            <div>
                <h2>Current Role</h2>
                <form action="{{ url_for('roles.run_analysis') }}" method="post" id="analysis-form">
                    <!-- <label for="cars">Choose a car:</label> -->
                    <select id="current_role" name="current_role" class="searchable-dropdown">
                    {% for role in roles %}
                        <option value="{{ role }}">{{ role }}</option>
                    {% endfor %}
                    </select>
                    <br><br>
            </div>
            <div>
                <h2>Current Salary</h2>
                    <input type="number" id="current_salary" name="current_salary" required>
            </div>
        </div>

    </div>

   <!-- Buttons Section (Run Analysis + Clear Results) -->
    <div class="buttons-container">
        <!-- Input Field and Run Analysis Button -->
            <button type="submit" class="action-button run-button" {% if not applicant %}disabled{% endif %}>
                Run Analysis
            </button>
        </form>

        <!-- Clear All Results Button -->
        <form method="post" action="/roles/clear" onsubmit="return confirm('Are you sure you want to delete all
uploaded resumes and clear results?');">
            <button type="submit" class="action-button clear-button" {% if not analysis_done %}disabled{% endif %}>
                Clear All Results
            </button>
        </form>
    </div>
    <!-- Analysis Results -->
    <div class="analysis-results">
        {% if resume_based %}
            <h2>Recommended Roles - Resume:</h2>
            <table class="table table-striped">
                <thead>
                    <tr>
                        {% for column in resume_based_col %}
```

```
                            {% if column == "annual_wage_variation" %}
                                <th>Expected Salary Change</th>
                            {% else %}
                                <th>{{ column.replace('_', ' ').title() }}</th>
                            {% endif %}
                        {% endfor %}
                    </tr>
                </thead>
                <tbody>
                    {% for row in resume_based %}
                        <tr>
                            {% for column in resume_based_col %}
                                {% if column == "annual_wage_variation" %}
                                    <td class="{% if row[column] > 0 %}positive{% else %}negative{% endif %}">
                                        {{ row[column] | usd }}
                                    </td>
                                {% elif column == "missing_skills" %}
                                    <td>{{ row[column] | join(', ') }}</td>
                                {% else %}
                                    <td>{{ row[column] }}</td>
                                {% endif %}
                            {% endfor %}
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
            <h2>Recommended Roles - Role:</h2>
            <table class="table table-striped">
                <thead>
                    <tr>
                        {% for column in resume_based_col %}
                            {% if column == "annual_wage_variation" %}
                                <th>Expected Salary Change</th>
                            {% else %}
                                <th>{{ column.replace('_', ' ').title() }}</th>
                            {% endif %}
                        {% endfor %}
                    </tr>
                </thead>
                <tbody>
                    {% for row in role_based %}
                        <tr>
                            {% for column in role_based_col %}
                                {% if column == "annual_wage_variation" %}
                                    <td class="{% if row[column] > 0 %}positive{% else %}negative{% endif %}">
                                        {{ row[column] | usd }}
                                    </td>
                                {% elif column == "missing_skills" %}
                                    <td>{{ row[column] | join(', ') }}</td>
                                {% else %}
                                    <td>{{ row[column] }}</td>
                                {% endif %}
                            {% endfor %}
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        {% endif %}
    </div>
</div> <!-- End of content-wrapper -->
<!-- Loading Spinner Overlay -->
<div id="loading-overlay">
    <div class="spinner"></div>
    <p>Running analysis, please wait...</p>
</div>
{% endblock %}

{% block scripts %}
<script>
    // Searchable dropdown from Select2
    $(document).ready(function() {
    $('.searchable-dropdown').select2({
        placeholder: "Select a role",
        allowClear: true
    });
    });
    // Show the loading overlay when the analysis form is submitted
    document.getElementById('analysis-form').addEventListener('submit', function() {
        document.getElementById('loading-overlay').style.display = 'block';
    });
</script>
{% endblock %}
```