Talent Recommendation Engine

1. Introduction

In today's highly competitive job market, connecting the right candidate with the appropriate job position is more critical than ever. For many companies, the manual evaluation of each applicant was the main screening procedure. While this method allowed for a more detailed view of a candidate's credentials, it was time-consuming and prone to human biases and errors. The quest for optimal alignment has necessitated the development of a Talent Recommendation Engine.

2. Previous Solutions

Historically, the problem of matching candidates to jobs has been tackled through various means:

1. **Keyword Matching:** This approach scanned resumes for specific words or phrases that aligned with the job description.

2. **Early Machine Learning Approaches:** Some previous solutions did utilize machine learning algorithms to predict a candidate's fit to a job. These models were generally based on feature extraction and the use of SVMs or decision trees to classify resumes.

These techniques frequently lacked the depth of analysis necessary to fully understand the complex nature of a candidate's fit for a position.

3. Problem Definition & Objectives

The challenge lies in creating an intelligent and automated process that can match candidates to job opportunities based on three factors: (1) language skills, (2) working experience, and (3) skills.

Hence, our main goal is to initially extract these three aspects from the resumes and create a ranking algorithm that can utilize them to arrange resumes in order of preference.

4. Experimental Setup

4.1. Dataset

The dataset encompasses a corpus of 13 unique job descriptions and an assortment of 126 resumes. Regrettably, two of these resumes were identified as containing data corruption, leading to the successful parsing and subsequent processing of 124 resumes.

4.2. Text Extraction

Our initial approach involved utilizing the PyPDF2 library. However, we identified a limitation in the PyPDF2 framework wherein it failed to maintain accurate text extraction order from resumes containing multiple columns. Consequently, we undertook a systematic exploration of various PDF parsing solutions, encompassing different iterations of PyPDF, PyMuPDF, PDFPlumber, and PDFMiner. Upon rigorous evaluation, PDFMiner emerged as the most proficient performer in

terms of text extraction accuracy. Further details about the text extraction and cleaning process will be discussed in the “Final Proposed Approach” section.

4.3. Initial Approach Using TF-IDF

Initially, a ranking approach was adopted based on the method described in Tejaswini et al.

This method was implemented using scikit-learn’s “TfidfVectorizer”. First, the job description’s and CV’s texts were extracted and cleaned, then vectorized using the “TfidfVectorizer” before finally computing a similarity score between the job description and each CV.

Although simple to implement, this TF-IDF method resulted in low accuracy (results shown in the Results section).

4.4. Parsing

In an attempt to extract important information from textual content, we initially explored conventional NLP methods.

The application of the spaCy NLP framework revealed limitations in grasping the contextual nuances even within rudimentary texts, as depicted in the provided illustration.

Figure 1. An image depicting the NLP analysis conducted using spaCy.

To illustrate, the spaCy model exhibited a deficiency in identifying English or Arabic, erroneously categorizing French as a "NORP" entity rather than a language.

We also explored pre-existing resume parsers, such as PyResParser. This tool processes PDF files and yields a dictionary containing the extracted information. However, it is noteworthy that even established resume parsers encountered challenges in accurately extracting information from CVs, as illustrated in the depicted figure below.

Figure 2. An image depicting the JSON dictionary derived through PyResParser

Evidently, a notable observation arises regarding the absence of crucial information as highlighted in yellow.

4.5. Exploration of Large Language Models (LLMs)

Recognizing the limitations of the TF-IDF method, manual parsing and conventional NLP methodologies, the focus shifted towards LLMs.

Table comparing the different LLM models in terms of computational requirements, accuracy and cost.

We initially opted to try various models such as Free Dolly, Llama 2 and Falcon. Eventually, OpenAI's ChatGPT stood out for accuracy and API ease. The results for each model's pros and cons are summarized in the table above.

4.6. Prompt Engineering

Prompt engineering was crucial for accurate data parsing. LangChain was utilized, with a response schema for preferred JSON formatting. Crafting a single prompt sufficed for job descriptions, yet CV parsing proved complex. Using one prompt caused inaccuracies and even hallucinations. Due to CV intricacies, employing multiple brief prompts yielded optimal accuracy. SequentialChain occasionally led to inconsistencies and JSON problems. Dividing the task into distinct prompts for specific CV aspects resolved this issue.

4.7. Embeddings

Word2Vec and GloVe embeddings were first considered to compute similarity scores. Nonetheless, their vocabulary is deficient in encompassing domain-specific terms such as

"Kubernetes" and "Kotlin," as well as words featuring punctuations like "C#," "C++," "Node.js," and "React.js." This limitation renders these models unsuitable for our objective. Eventually, we utilized the ADA-002 text embedding due to its precision, extensive vocabulary, and cost-effectiveness.

5. Final Proposed Approach

Below, we detail the various components of our proposed approach:

5.1 Data Preprocessing

- **Text Extraction:** We exclusively focused on job descriptions and resumes stored in the docx and pdf formats. In the case of docx files, we employed the python-docx library, while for PDF files, we selected the PDFMiner option (cf. section 4.2).

- **Text Cleaning:** The cleaning process encompasses two stages: Text Validation and Unwanted character removal.

    - **Text Validation:** During the text validation process, our focus lies in verifying the success of text extraction. Our initial assessment entails determining whether the extracted text attains a minimum threshold of characters or words. Additionally, our assessment encompasses the identification of undesired line breaks occurring between word characters. Our text validation process ensures the removal of inadvertently introduced line breaks, thereby preserving text coherence and integrity.

    - **Unwanted character removal:** In a subsequent stage, the validated text undergoes a cleaning process aimed at removing non-printable characters, exemplified by symbols like "•", as well as multiple consecutive spaces and line breaks. Notably, we choose to retain punctuation marks, such as commas and dots, as they are perceived to carry contextual significance and positional information that contributes to potential meaning enhancement and augments the precision of text processing.

An example of a text before and after cleaning

5.2. Parsing

Our engine makes use of LangChain with OpenAI's ChatGPT 3.5 Turbo. Using LangChain, we can define the output schema of the responses, which allows us to parse the extracted information in a well-defined and structured format.

The saved JSON files for job descriptions and resumes are structured as follows:

- **Parsing Job Descriptions:** We crafted a specialized prompt to extract key components from the job descriptions, including the job title, total years of experience required, languages needed, and required skills. The extracted information is then saved into a structured JSON file.

- **Parsing Resumes:** Recognizing the complexity of resumes and their different formats, we approached the task in a step-wise manner, employing multiple simple prompts for various aspects.
    - **Experience Breakdown**: Sequential prompts were used to simplify the resume into job titles and dates, disregarding organization names, and further translating the dates for each job title into durations.
    - **Skills Extraction:** Sequential prompts extracted the skills from the entire resume, followed by a cleaning process to align the skills with the industry-specific requirements.
    - **Language Extraction**: A designated prompt extracted language skills and proficiencies.
    - **Combining Parsed Elements**: The individual parts were then combined into one single JSON file, representing the parsed CV of the candidate.

By leveraging LLMs and implementing a structured and sequential parsing strategy, our system ensures precise extraction of the relevant information.

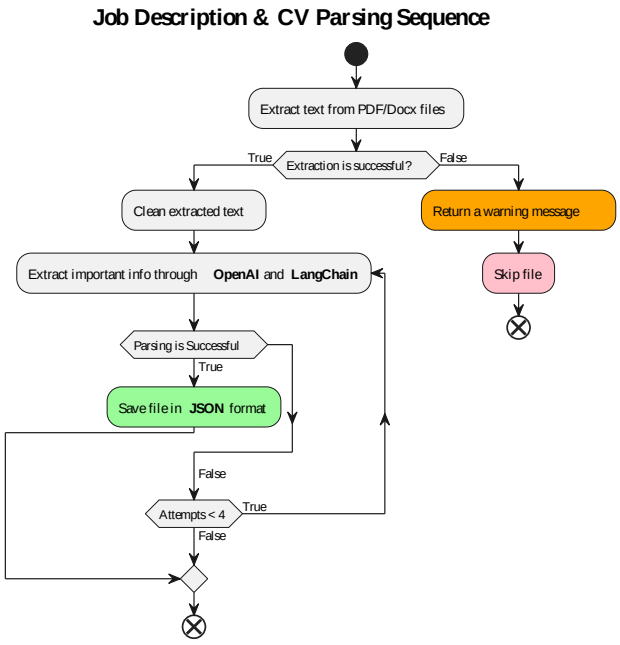

Figure 3. Flow chart showing the parsing sequence.

5.3. Ranking

The ranking process involves a series of distinct steps, outlined in the accompanying chart.

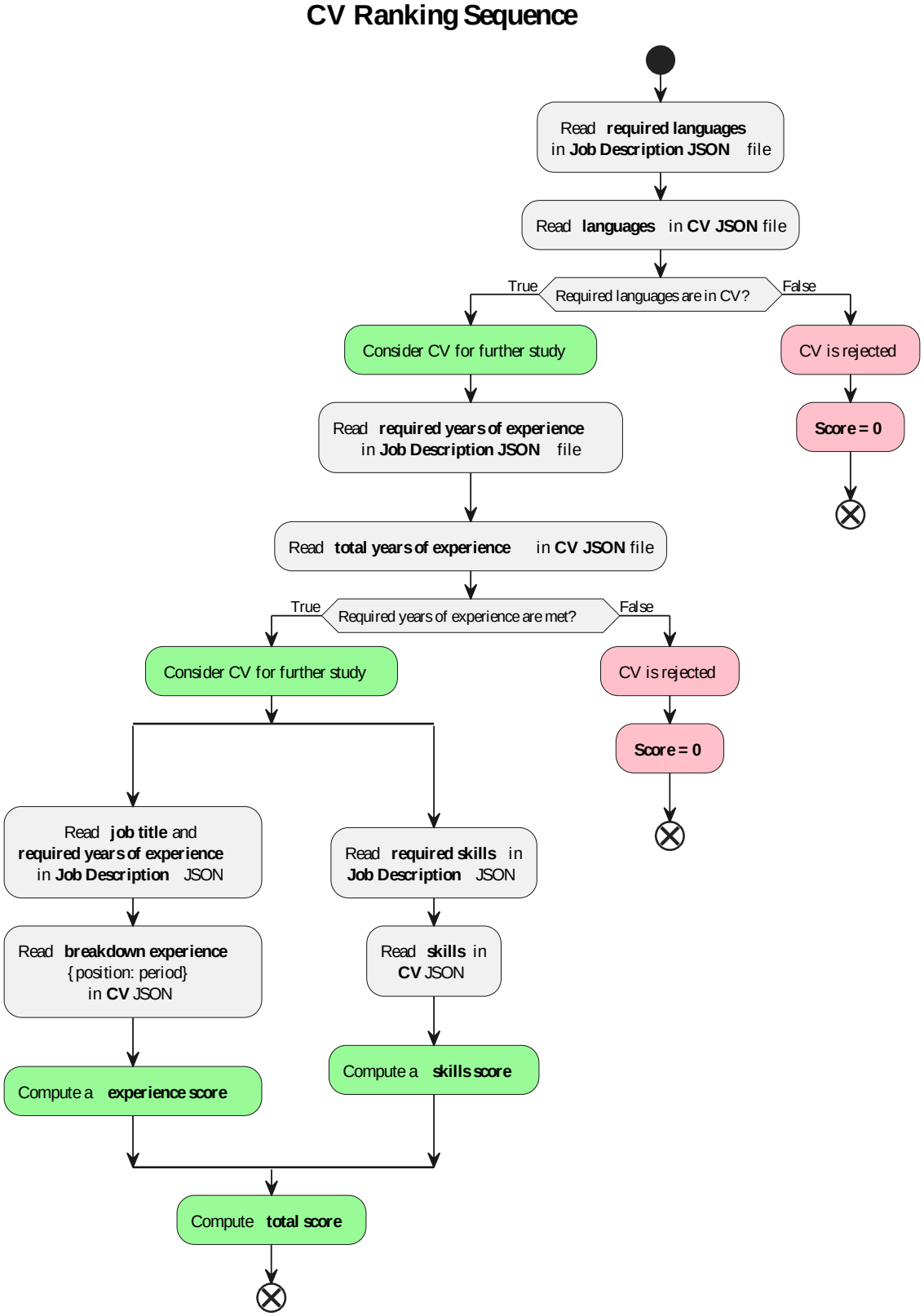

Figure 4. Flow chart showing the ranking sequence.

First, we extract the requisite language proficiencies from the job description JSON file and cross-reference them with the spoken languages specified in the resume, also stored in a JSON file. Should the language criterion be satisfied, the CV is deemed eligible for further consideration.

Two distinct language assessment modes, namely "strict" and "relaxed," have been developed for this purpose.

- **Language Assessment Modes:** Language proficiency is evaluated on a scale from 0 to 5. A candidate is deemed to possess language proficiency if their proficiency level exceeds 2.
    - ***strict*** mode: the candidate should be proficient in all the required languages stated in the job description.
    - ***relaxed*** mode: the candidate should be proficient in at least two-thirds of the required languages stated in the job description.

If the CV passes the language assessment, we proceed to extract the minimum years of experience from the job description JSON file and the breakdown experience and total years of experience from the CV JSON file. If the candidate passes the years of experience assessment, then the resume undergoes the evaluation process.

This evaluation process encompasses assigning a score based on the candidate's work experience and skills, relative to the job title and the essential skills outlined in the job description.

- **Skill scoring modes:**

Two skill scoring modes have been developed: Match and ADA.

- ***Match***: This mode adopts a fuzzy matching strategy, where strings are deemed a match if their similarity exceeds 80%. This approach effectively addresses certain instances, such as “Node”, “NodeJS”, and “Node.js”, while acknowledging its imperfection, as illustrated by the non-matching of “javascript” and “js”.
- ***ADA***: This mode utilizes the "text-embedding-ada-002" embedding model provided by OpenAI. Two embeddings, one for the job description skills and another for the CV skills, are generated and a cosine similarity score is computed.

- **Work experience scoring modes:**

Three Work experience scoring modes have been developed: Match, ADA, and ADAMa.

- ***Match***: This mode bears some resemblance to the Match mode employed for skills. However, it distinguishes itself by employing exact matching on segments of the string to achieve the intended outcome. For instance, "DevOps Engineer" will yield a positive match with "DevOps/DBA Engineer," and "Junior DevOps Engineer."

- ***ADA***: This mode bears some resemblance to the ADA mode employed for skills. However, instead of computing the cosine similarity for skills, we compute the cosine similarity for job titles/positions

- ***ADAMa***: This mode combines attributes from both the ADA and Match modes, thus bearing the name ADAMa. It offers an augmented mode in which the user aids the algorithm in enhancing ranking accuracy. A cosine similarity is computed for job profiles using the ada-002 model. Subsequently, the user provides a list of desired job titles for prioritization. If these titles are identified within the experience breakdown, the CV receives a score boost proportional to the years of relevant experience.

5.4.Streamlit

Using the Streamlit framework, we have constructed an intuitive user interface facilitating seamless interaction with the ranking engine. This interface allows users to effortlessly upload CVs and job descriptions, along with the ability to swiftly switch between distinct ranking modes through a streamlined button-click process.

6. Results and Discussion

Job Description and CV JSONs

One of the major outcomes of this project resides in the job description and CV parsing. Visual representations of exemplary job description and CV documents, alongside corresponding JSON files, are presented in the images below.

DevOps Engineer - Infrastructure Focus

At Turning Technologies, our mission is to enable engaging and interactive experiences that improve learning. We are a world leader in audience response technologies ranging from a variety of hardware clickers to our innovative SaaS based web platform.

Responsibilities

1. Maintain, modify, and enhance new and existing application infrastructure
2. Work with development teams for architecture design and creation of requirements
3. Evaluate new and emerging technologies for implementation
4. Create robust, scalable, and highly-available infrastructure for a variety of applications
5. Work collaboratively with team members

Abilities

1. Thrive in a highly collaborative team environment
2. Capable of effectively communicating technical concepts to team members
3. Demonstrate aptitude for learning new technologies
4. Have a positive attitude and thrive in a dynamic environment
5. Exhibit strong interpersonal and communication skills (both verbal and written)
6. Possess strong analytical and problem-solving skills

Qualifications

1. Well versed in DNS, Load Balancing, SSL, TCP/IP, networking and security
2. Solid scripting skills (e.g. Python / Java, shell scripting)
3. Database experience (e.g. MySQL, Postgres)
4. Experience using source control such as GIT, SVN or equivalent

Focused Qualifications

1. Bachelor's Degree in Computer Science or related field
2. 1-3 years of professional cloud-based infrastructure (AWS Preferred)
3. 1+ years of experience with Java development
4. 1+ years of experience with SQL database design and development
5. 1+ years of experience working in an agile environment
6. Familiarity with web service design and development

Figure 5. Example of a job description


```
{
    "job_title": "DevOps Engineer",
    "total_years_required": 1,
    "languages": [
        "English"
    ],
    "skills_required": [
        "DNS",
        "Load Balancing",
        "SSL",
        "TCP/IP",
        "Networking",
        "Security",
        "Python",
        "Java",
        "Shell Scripting",
        "MySQL",
        "Postgres",
        "GIT",
        "SVN",
        "AWS",
        "SQL Database Design",
        "Agile Methodology",
        "Web Service Design"
    ]
}
```


Figure 6. Example of its corresponding JSON file

Figure 7. Example of a resume

Figure 8. Example of its corresponding JSON file

In the depicted visual representations, we accentuate the acquired information in both the JSON files and the original source file using matching colors. As observed in the figures, the ChatGPT/LangChain model facilitates an exceedingly precise extraction of important information from both job descriptions and CVs.

Regarding the job description content, the model adeptly identifies and retrieves the job title as "DevOps Engineer," accurately assesses the required years of experience as "1" year, and successfully captures the specified array of requisite skills.

Concerning the CV content, our analysis reveals the model's ability to accurately discern and retrieve the candidate's job roles, such as "Software Engineer," along with their corresponding durations, exemplified by "11.08" years. Furthermore, the model proficiently captures the languages "Arabic" and "English," accompanied by their respective proficiency scores, notably "5" for native and "4" for full professional. The model also demonstrates a high level of success in recognizing most skills.

In comparison with outcomes yielded by prior models like spaCy and PyResParser, the approach adopted herein signifies a remarkable milestone in the domain of resume parsing.

7. Rankings

We were provided with the resume rankings for 3 job descriptions as follows:

(It is important to note that these rankings are also post-interview.)

DevOps Engineer	Dotnet Developer	Quantitative Developer
Resume	Resume	Resume
#1 Elie Youssef	#1 Bernard Estephan	#1 Tony Mattar
#2 Hussein Hijazi	#2 Mohammad Darweech	#2 Patricia Boulos
#3 Charbel Nakad		

Table 1. Rankings as provided by Supportful

TF-IDF

Initially, we executed the rankings employing the previously outlined TF-IDF method to establish a reference benchmark. The outcomes of the ranking procedure are presented within the subsequent tables. A color-coded scheme has been implemented for enhanced readability of the table rows.

Hereafter, rows highlighted with a red shade indicate a lack of alignment between the profile and the job description, whereas a green shade signifies alignment. Notably, the color yellow has been designated to underscore the ranking of candidates selected by the company.Rank	Resume
Rank 16	Hassan El Hajj CV.pdf
Rank 25	Hussein_Hijazi_CV.pdf
Rank 35	Elie+Youssef+CV.pdf
Rank 41	Charbel Nakad CV.pdf

Table 2. DevOps Engineer rankings with TF-IDF

As observed in the table above, the three candidates chosen by the company do not appear in the top 20 list. Meanwhile, the highest-ranked CV that matches the job description holds the 16th position. The TF-IDF approach does not yield a satisfactory ranking result for the DevOps Engineer scenario.

Rank	Resume
Rank 18	Charbel Beainy CV.pdf
Rank 61	Bernard Estephan CV.pdf
Rank 78	Mohammad Darweech CV.pdf

Table 3. DotNet Developer rankings with TF-IDF

As seen in the table provided, the company's selected candidates hold ranks of 61 and 78. Meanwhile, the highest-ranked CV that matches the job description holds the 18th position. Notably, the TF-IDF approach exhibits even weaker performance in the case of the DotNet Developer scenario.

Rank	Resume
Rank 29	Abbas Khansa CV.pdf
Rank 75	Tony Mattar CV.pdf
Rank 79	Patricia Boulos resume.pdf

Table 4. Quantitative Developer rankings with TF-IDF

As shown in the table, the company's chosen candidates have rankings of 75 and 79. In contrast, the top-ranked CV that aligns with the job description is placed 29th. Furthermore, the TF-IDF method shows limited effectiveness in the case of the Quantitative Developer scenario.

In summary, the TF-IDF technique performed poorly in CV ranking. This inadequate outcome is primarily due to the technique treating the CV as a single entity, without analyzing its distinct sections and applying individual ranking methods to each.

ADA

Next, we proceed to showcase the rankings achieved through our ADA methodology.

Rank	Resume
Rank 1	Hussein_Hijazi_CV.json:
Rank 2	Hassan El Hajj CV.json:
Rank 3	Sami Abou Jaoude CV.json:
Rank 4	Charbel Nakad CV.json:
Rank 5	Amin Hassoun CV.json:
Rank 6	Tarek Srour CV.json:
Rank 7	Elie+Youssef+CV.json:
Rank 8	Ali Tarhini CV.json:
Rank 9	Ali Rahal CV.json:
Rank 10	Elie Abi Khalil CV.json:

Table 5. DevOps Engineer rankings with ADA

As seen in the table provided, the three candidates selected by the company are already positioned within the top 10 rankings, holding ranks 1, 4, and 10. Moreover, among the overall top 10, a total of 8 CVs align closely with the job description. This outcome exhibits a significant enhancement when compared to the TF-IDF approach.

Rank	Resume
Rank 1	Amin Atwi CV.json:
Rank 2	Charbel Kahwaji resume.json:
Rank 3	Othman Al Moussawel
Rank 4	Malak Daher CV.json:
Rank 5	Ali Tarhini CV.json:
Rank 6	Toufic Sleiman CV.json:
Rank 7	Tony Mattar CV.json:
Rank 8	Roc Khalil CV.json:
Rank 9	George_Abboud_CV.json:
Rank 10	Youssef Kanso CV.json:
Rank 16	Bernard Estephan CV.json:
Rank 23	Mohammad Darweech

Table 6. DotNet Developer rankings with ADA

Regarding the DotNet Developer role, the table illustrates lower performance compared to the DevOps position. The two candidates selected by the company hold ranks 16, and 23. This marks a significant enhancement from the TF-IDF method, which positions them at 61 and 78. Within the top 10 overall, 4 CVs match the job description. The diminished performance can primarily be attributed to the experience score, as none of the candidates' resumes contain a position as a DotNet Developer, in contrast to the case of DevOps Engineer.

Rank	Resume
Rank 1	Habib Al Khoury-CV.json
Rank 2	Nour Khouja CV.json
Rank 3	George_Abboud_CV.json
Rank 4	Tony Mattar CV.json
Rank 5	Ali Tarhini CV.json
Rank 6	Abbas Khansa CV.json
Rank 7	Othman Al Moussawel_Resume.json
Rank 8	Wissam Assaf CV.json
Rank 9	Bernard Estephan CV.json
Rank 10	Ibrahim Nour CV.json
Rank 29	Patricia Boulos resume.json

Table 7. Quantitative Developer rankings with ADA

Regarding the Quantitative Developer role, the table again illustrates lower performance compared to the DevOps position. The two candidates selected by the company hold ranks 4, and 29. This again is a significant improvement from the TF-IDF method, which positions them at 75 and 79. Within the top 10 overall, 3 CVs match the job description. The diminished performance once again can primarily be attributed to the experience score, as none of the candidates' resumes contain a position as a Quantitative Developer, in contrast to the case of DevOps Engineer.

In summary, the ADA technique showed a drastic increase in accuracy as compared to the TF-IDF technique. However, it showed a lower performance for DotNet and Quantitative Developer due to the fact that these position do not exist in any of the candidates' CVs.

ADAMa

To address the challenge of reduced accuracy, we created a hybrid ADA and Keyword Matching approach called ADAMa. In essence, ADAMa is an ADA technique with user involvement, wherein the user selects a set of prioritized job titles for ranking.

The ADAMa method shows a slightly improved accuracy compared to ADA for the role of DevOps Engineer. The company's three chosen candidates remain in the top 10 rankings. Additionally, all 10 CVs now match the job description perfectly.

Rank	Resume
Rank 1	Michel Atallah CV.json:
Rank 2	Hussein_Hijazi_CV.json:
Rank 3	Tarek Srour CV.json:
Rank 4	Hassan El Hajj CV.json:
Rank 5	Johnny Zghaib Resume.json:
Rank 6	Amin Hassoun CV.json:
Rank 7	Charbel Nakad CV.json:
Rank 8	Elie+Youssef+CV.json:
Rank 9	Elie Abi Khalil CV.json:
Rank 10	Raghid Zgheib CV.json:

Table 8. DevOps Engineer rankings with ADAMa

Rank	Resume
Rank 1	Amin Atwi CV.json
Rank 2	Wissam Assaf CV.json
Rank 3	Mohammad Darweech CV.json
Rank 4	Ibrahim Nour CV.json
Rank 5	Charbel Kahwaji resume.json
Rank 6	Reda Tabeja CV.json
Rank 7	Bernard Estephan CV.json

Rank 8	Charbel Beainy CV.json
Rank 9	Mohamad Khachab CV.json
Rank 10	Samer Dernaika CV.json

Table 9. DotNet Developer rankings with ADAMa

As for the DotNet rankings, the two candidates selected by the company hold ranks 3, and 7. An improvement from the previous 4, and 29. The used keywords for this enhacement are "Fullstack", "Frontend", "Backend", "Software". Within the top 10 overall, 5 CVs match the job description. A slightly improved performance.

Rank	Resume
Rank 1	Habib Al Khoury-CV.json
Rank 2	Ali Rahal CV.json
Rank 3	Ali Tarhini CV.json
Rank 4	
Rank 5	George_Abboud_CV.json
Rank 6	Malak Daher CV.json
Rank 7	Ali Dandach CV.json
Rank 8	Jana Khanji CV.json
Rank 9	Patricia Boulos resume.json
Rank 10	Tony Mattar CV.json

Table 10. Quantitative Developer rankings with ADAMa

For the Quantitative Developer rankings, the two candidates selected by the company hold ranks 9, and 10. An overall improvement from the previous 4, and 29. The used keywords for this enhacement are "Fullstack", "Frontend", "Backend", "Software". Within the top 10 overall, 4 CVs match the job description. A slightly improved performance.

In summary, the ADAMa technique showed a further slight increase in accuracy as compared to the ADA technique. However, its overall accuracy relies a lot on the quality of the job descriptions and the CVs.

Conclusion and Future Improvements

In conclusion, our method initially involved developing a remarkably precise job description and resume parser. Notably, the parsing accuracy is exceptionally high and surpasses that of traditional parsers.

The results from the rankings demonstrated a significant enhancement compared to the TF-IDF method employed in literature. The accuracy achieved with our small test sample reached 63%. This low accuracy can be attributed primarily to two main factors: Data and Embeddings.

Future improvements should explore:

- **Data**: We were given a set of 128 CVs and 13 job descriptions for analysis. Although this data is a good starting point, future enhancements should involve using well-organized and cleaned

data. This data should include a distinct ranking for each CV and a scoring system. Additionally, a comprehensive chart depicting various computer language skills and their importance for each job title would be beneficial.

- **Embeddings**: In our approach, we utilize OpenAI's ada-002 embeddings. Although ada-002 is effective for semantic search, it has limitations as it is trained on the entire English vocabulary and may not fully capture the subtleties of computer skills. Future improvements should seek the fitting and training of a task-specific tokenizer and embedders.

- **LLMs**: In our method, we utilize OpenAI's ChatGPT. Although the model is very accurate, it tends to be relatively slow during the parsing process. A more effective strategy could involve refining a Large Language Model (LLM) specifically for the task of resume parsing.