AUTOMATED SKILL EXTRACTION AND JOB MATCHING USING ADVANCED LANGUAGE LEARNING MODELS

BERNARD A. GEORGES AND ALICE TEIXEIRA

ABSTRACT. In the evolving landscape of recruitment, efficiently identifying and evaluating candidate skills from resumes is crucial for both employers and job seekers. This project aims to develop an application using a Language Learning Model (LLM) designed to automate the extraction of core skills from candidate CVs. Leveraging advanced natural language processing (NLP) techniques and machine learning algorithms, the application will analyze the unstructured data within resumes to accurately identify and categorize pertinent skills. By implementing this application, we anticipate significant improvements in the job searching process, including reduced time-to-hire, enhanced candidate matching, and streamlined HR workflows. Furthermore, after extracting the skills, the application will use this information to find and suggest matching job opportunities. This innovation promises to transform traditional resume screening methods, offering a scalable and reliable solution to skill identification and job matching in the recruitment domain.

1. INTRODUCTION

In the fast-paced world of recruitment, the efficiency and effectiveness of the initial resume screening process can significantly impact the overall hiring timeline and quality of candidate matches. On average, a recruiter spends only about 10 seconds reviewing a CV, which underscores the challenge of thoroughly identifying and evaluating the core skills of candidates in such a limited time. This often leads to potential mismatches and missed opportunities, both for employers seeking the right talent and for job seekers aspiring to find suitable positions.

To address this critical challenge, our project aims to develop an application leveraging a Language Learning Model (LLM) designed to automate the extraction of core skills from candidate CVs. By employing advanced natural language processing (NLP) techniques and machine learning algorithms, the application will analyze the unstructured data within resumes to accurately identify and categorize pertinent skills..

Moreover, the application will go beyond skill extraction by using the identified skills to find and suggest matching job opportunities. This dual functionality not only enhances the recruitment process by reducing time-to-hire and improving candidate matching but also streamlines HR workflows, providing a scalable and reliable solution for both recruiters and job seekers in the dynamic recruitment landscape.

2. STATE OF THE ART

The use of AI in the job searching market is a already common practice, with several platforms like LinkedIn and Glassdoor as well as several papers like *JobRecoGPT: Explainable job recommendations using LLMs*[1] exploring that reality and the benefits they bring.

However, their integration is not without any drawbacks. Whenever similar technology is utilized, concerns are raised regarding privacy, especially when working with personal, identifiable data, as well as ethical considerations and transparency, considering the black-box models used.

In the work we propose, transparency is mitigated through code, as we can visualize the words the LLM selects in order to then make a compilation of job listings.

3. Methodology

As with any application, this project can be divided into multiple components. Starting with the front-end, users will have the capability to upload their PDF CVs or even a brief cover letter highlighting their skills. The back-end then takes over, responsible for extracting, cleaning, and processing the text submitted through the front-end. This structured approach ensures a seamless and efficient workflow from user input to skill extraction and job matching.

3.1. **Front-end.** Due to some time restrictions and some difficulty in the back-end the front-end was kept simple. Developed in Python, by the use of the library Flask, a lightweight web framework for Python that allows the creation of web applications quickly and easily. It is designed to be simple and flexible, providing the essential tools needed to build a web app without imposing many restrictions. The front-end is composed of 3 main pages.

3.1.1. *Home Page.* Initially intended to demonstrate the product or all available jobs, this was page was later scraped being left as place holder as a future improvement for the page. From this a user can access the pages to load its cover letter or CV through the side panel openable through the top left.

3.1.2. *Cover Letter Page.* This page is intended for users that wish to search through the use of a cover letter, a small text containing some motivations, what the candidate is looking for and some past experiences. When submitted it is redirected directly to the model that pre-processes and extracts the information passed. This is a more efficient and faster then the pdf as it normally contains less information in a more condense and structured way. The example used to test is shown bellow.

> Dr. John Smith is a senior researcher in Artificial Intelligence at MIT. He holds a PhD in Computer Science from Stanford University. Previously, worked at Google in California.

Figure 1. Cover Letter used to test the model

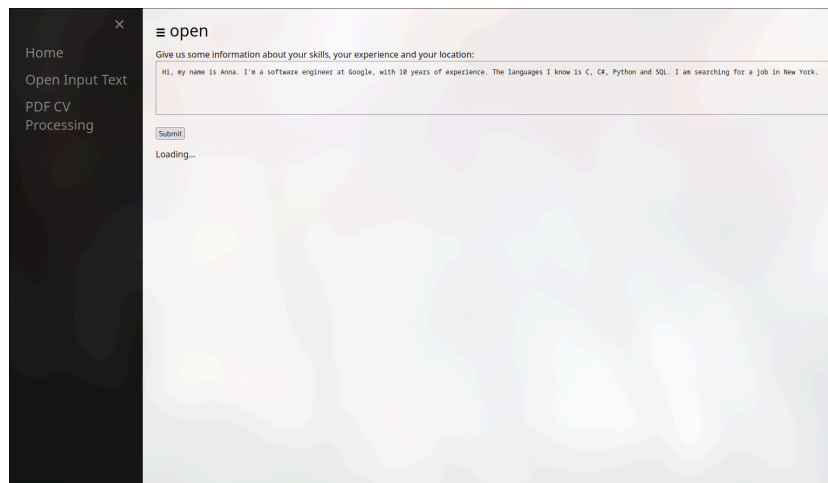FIGURE 2. Web Page intended for the insertion of the cover letter

FIGURE 3. Web Page for cover letter insertion - loading while processing data
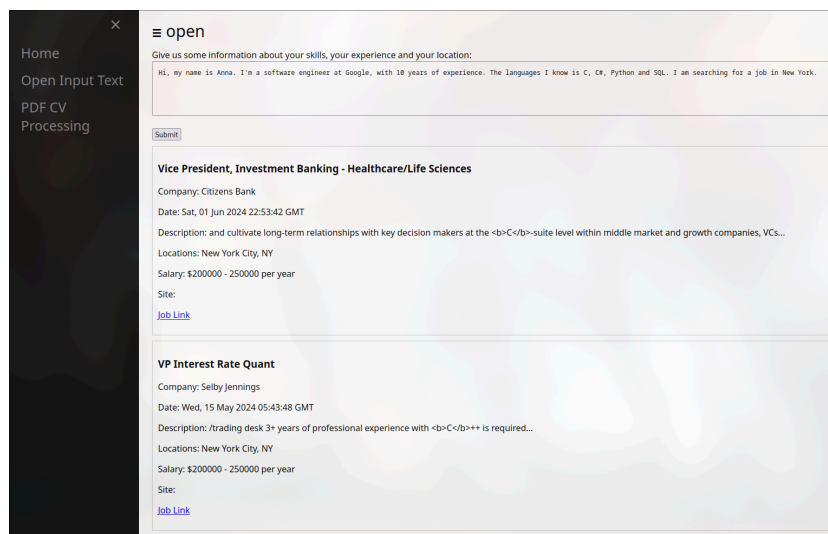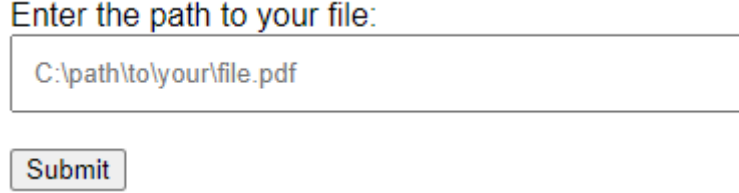

FIGURE 4. Web Page for cover letter insertion - displaying results

3.1.3. *CV Page.* Finally, the page intended for a PDF format. In this page the user enters the path to the intended CV. When submitted the front-end first reads the file only then sending the extracted text to the model. Although slower, this method allows for a more thorough job search as it contains all experiences and skills.

Enter the path to your file:

C:\path\to\your\file.pdf

Submit

FIGURE 5. Web Page intended for the insertion of the path of a PDF file

3.2. **Back-end.** The most complex part of the development is constructing the back-end, which begins with the critical decision of selecting the appropriate Language Learning Model (LLM). With the increasing popularization of machine learning and the growing number of LLMs available to the public, there are numerous options to consider, each with unique capabilities and potential applications. This choice will significantly impact the performance and effectiveness of the application in extracting and processing skills from CVs.

3.2.1. *Falcon 7B.* The Falcon LLM model, were the first models to be considered as they present several advantages and disadvantages for this project. On the positive side, Falcon LLM is known for its high accuracy in understanding and generating human-like text, making it well-suited for nuanced tasks such as skill extraction from CVs. Its design allows it to handle a wide range of text inputs, ensuring robustness across various CV formats and industries. By leveraging state-of-the-art natural language processing techniques, Falcon LLM excels in capturing context and semantics, which is crucial for accurately identifying skills.

However, there are notable disadvantages. Falcon LLM requires significant computational resources. Ultimately this requirement unabled the use of this model as the computers used by the group could not withstand the needs.

3.2.2. *BERT.* The BERT (Bidirectional Encoder Representations from Transformers) model, in theory, presents several advantages and disadvantages for this project. BERT is renowned for its ability to capture bidirectional context in text, making it highly effective in various natural language processing tasks, including semantic understanding and information extraction from documents like CVs. Its pre-trained representations have been shown to achieve state-of-the-art performance on a wide range of NLP tasks.

However, during the evaluation phase, BERT demonstrated subpar results in skill extraction from CVs compared to other models specifically designed for this task. Therefore, despite BERT's strong performance in general NLP tasks, its subpar results in skill extraction from CVs led to the decision to explore other models better suited for this specific application. This highlights the importance of choos-

ing a model that is not only powerful but also specifically optimized for the task at hand to achieve the desired level of accuracy and efficiency.

3.2.3. *Together API.* Ultimately, selecting a model that delivers superior results while remaining compatible with the group's computational resources was essential. While several high-performing models were considered, many were paid models, which could lead to increased costs over time. To mitigate this expense, the group opted to use the Together API, which offers a free trial period. This approach allowed the team to utilize many advanced models without incurring significant costs. After some searching there was the choice of using the mistral model

3.2.4. *Prompt Pre-processing.* To improve the accuracy and efficiency of the models, a prompt was created to clearly define the system's objectives. This was accomplished using the LangChain library to generate a prompt template, which included the system's goals and incorporated the input text.

Additionally, a pre-processing step was introduced. This step involves removing URLs, email addresses, and dates from the CV or text. When dealing with PDF formats, the text must first be extracted from the file before performing this pre-processing. This ensures that the text fed into the models is clean and free of irrelevant information, thereby enhancing the overall quality of skill extraction.

```python
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "You are an expert extraction algorithm. "
            "Only extract relevant information from the text. "
            "If you do not know the value of an attribute asked to extract, "
            "Never return the examples placeholder, try to use it as a reference only."
            "Return null for the attribute's value.",
        ),
        # Please see the how-to about improving performance with
        # reference examples.
        # MessagesPlaceholder('examples'),
        ("human", "{text}"),
    ]
)
```

FIGURE 6. Prompt template created

The code above helps to instruct the algorithm to extract only relevant information, handle unknown attribute values by not returning placeholder examples, and return null if the attribute's value is unknown. The template also reserves a placeholder for the user's input, which will replace the {text} placeholder.

3.2.5. *Result Templating.* Additionally, a results template was created using LangChain to help structure the model's output. This code snippet defines a class `Person` using the BaseModel from Pydantic, providing a schema for extracting information from a candidate's CV. Each field in the `Person` class is optional and comes with a detailed description, which is sent to the LLM to improve extraction results.

This approach offers several advantages. The class `Person` clearly defines the schema for the data to be extracted, including the candidate's name, skills, experiences, and locations, helping the model understand what information to look for and how to categorize it. By making each field optional, the model gains flexibility,

as it is not forced to extract every piece of information if it's not available, thus enhancing accuracy. Detailed descriptions for each field guide the LLM in identifying and extracting the relevant data accurately, significantly improving the quality of the extracted information.

Moreover, providing a structured schema with descriptions improves the model's ability to extract precise information from unstructured text, leading to more reliable and consistent results. The use of doc-strings and inline comments makes the code easier to understand and maintain, ensuring that future modifications can be made with a clear understanding of the current setup. By using this structured approach, the model can deliver well-organized and relevant results, enhancing the overall efficiency and effectiveness of the skill extraction process from CVs.

```python
class Person(BaseModel):
    """The skill from a candidate's CV."""

    # " Doc-string for the entity Person.
    # This doc-string is sent to the LLM as the description of the schema Person,
    # and it can help to improve extraction results.

    # Note that:
    # 1. Each field is an `optional` -- this allows the model to decline to extract it!
    # 2. Each field has a `description` -- this description is used by the LLM.
    # Having a good description can help improve extraction results.
    name: Optional[str] = Field(default=None, description="The name of the person")
    skills: Optional[list] = Field(default=None, description="The hard/soft skills acquired by the candidate")
    experiences: Optional[list] = Field(default=None, description="level of experience in a particular field, so any job/company or academic experience.")
    locations: Optional[list] = Field(default=None, description="The geografical location of where the person has worked or studied, any country, city, or region, dont give any university or company.")
```

FIGURE 7. Base Model used to template the resulting structure

```
Skills: ['Artificial Intelligence', 'Computer Science', 'Research']
Experiences: ['Senior Researcher', 'PhD Holder']
Locations: ['MIT', 'Stanford University', 'Google']
```

FIGURE 8. Result given by the model given the Figure 6

3.2.6. *Job search.* The data display on the job listings comes from Careerjet API. Communications with this api pressupose the entering of the user's public ip as well as data on the user agent, alongside keywords to be searched on the job listings and a location. The output of json from the API has an object with keys 'jobs', 'hits', 'response_time', 'type' and 'pages'. The value of the key 'jobs', is a list of objetcs with the keys: 'salary_min', 'locations', 'salary_type', 'date', 'description', 'salary_currency_code', 'salary', 'site', 'title', 'salary_max', and, lastly, 'company'.

That being said, only part of that data was displayed per job listing; namely: the title, company, date of the posting, description, location, salary and site.

4. RESULTS

The results was the display of job listings based on an input text or the processing of a CV, by processing keywords and locations.

Through tests we noticed difference results to the same input, a case that was forming due to different word selections by the LLM on the input data. Differing selections affected the quality of output in job listings, and thus we mitigated it a bit by selecting only the words the LLM deemed as location if there were a locational point (ie. a country, district, continent, etc.).

This alteration stabilized a bit the results, and as thus, produced better job listings through the API.

5. FUTURE WORK

There are several key areas that can be tested to further enhance the accuracy and efficiency of skill extraction from CVs. Firstly, fine-tuning the Mistral or any other model on domain-specific datasets could be crucial. Fine-tuning presents both challenges and advantages. One of the main challenges is the need for a significant amount of high-quality, labeled data specific to the domain, which can be time-consuming to collect. Additionally, the process requires substantial computational resources and expertise in machine learning to adjust the model's parameters effectively. However, the advantages of fine-tuning are considerable. It can significantly enhance the model's performance by tailoring it to recognize the nuances and specific terminology used within a particular field, leading to more accurate and relevant skill extraction. This customization can make the model more adept at handling the specific characteristics of CVs within the targeted domain, ultimately improving the overall efficiency and effectiveness of the extraction process.

Additionally, implementing a double-checking mechanism by using two models in tandem could significantly increase the reliability of the extracted information. For instance, outputs from the Mistral model could be cross-validated with another high-performing model to ensure consistency and accuracy. One of the primary disadvantages is the increased computational overhead, as running two models simultaneously requires more processing power and resources, potentially leading to higher costs and longer processing times. Moreover, integrating and coordinating the outputs of two models can add complexity to the system, necessitating more sophisticated error-handling and reconciliation mechanisms.

However, the advantages of a double-checking mechanism are substantial. It can greatly enhance the reliability and robustness of the extracted information, as discrepancies between the models' outputs can highlight areas of uncertainty and complement the answers, prompting further review or adjustment. This method can also provide a more comprehensive analysis by leveraging the strengths and compensating for the weaknesses of each individual model. By cross-validating results, the system can achieve higher accuracy and confidence in the extracted skills, ultimately leading to more reliable and valuable insights for the end-users.

Finally, exploring and testing more models will remain a priority. Although the LLaMA model was initially considered, certain adjustments are required to optimize its performance for this specific task. Continuing to evaluate and compare different models will help identify the most effective solutions for various aspects of the skill extraction process.

References

1. Preetam Ghosh, V. S.: JobRecoGPT: Explainable job recommendations using LLMs, (2023)

Department of Computing, University of Minho, Braga, Portugal

Department of Computing, University of Minho, Braga, Portugal