T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

UTILIZING LANGUAGE MODELS AND RAG SYSTEMS IN CANDIDATE SELECTION

ASUMAN SARE ERGÜT

SUPERVISOR
ASSISTANT PROFESSOR DR. BURCU YILMAZ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

UTILIZING LANGUAGE MODELS AND RAG SYSTEMS IN CANDIDATE SELECTION

ASUMAN SARE ERGÜT

SUPERVISOR
ASSISTANT PROFESSOR DR. BURCU YILMAZ

2024
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 21/01/2023 by the following jury.

JURY

Member
(Supervisor) : Assistant Professor Dr. BURCU YILMAZ

Member : Associate Profesor HABİL KALKAN

ABSTRACT

Recruiters have to review numerous resumes when a new candidate has to be recruited. Examining the data in each resume and analyzing the applicant's fit for the position is a complex and time-consuming task. Consequently, manual methods are being replaced by more optimized and high-performance software every day. The goal of this project is to assess best suited candidate for the job among other candidates. This is accomplished through the use of large language model and natural language processing techniques.

Keywords: resume, large language model, natural language processing, human resources.

ÖZET

İşe alım uzmanları, bir kişiyi/kişileri işe almaları gerektiğinde çok sayıda özgeçmişi gözden geçirmek zorundadırlar. Her özgeçmişte bulunan verileri incelemek ve bu verilerin alınacak işe uygunluğunu analiz etmek zor ve uzun süren bir iştir. Bu nedenle manuel yöntemler, yerini her geçen gün daha optimize ve yüksek performanslı yazılımlara bırakmaktadır. Bu proje, büyük dil modelleri ve doğal dil işleme tekniklerinin kombinasyonuyla başvuran adaylar arasından işe en uygun adayı bulmayı amaçlamaktadır.

Anahtar Kelimeler: özgeçmiş, büyük dil modeli, doğal dil işleme, insan kaynakları.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my instructors from whom I received the most help and support: Burcu YILMAZ, my consultant who carried out this project, Habil KALKAN, an excellent consultant for all types of engineering and Başak BULUZ KÖMEÇOĞLU, my supporter and inspiration.

Also, I would like to thank my parents, my fiancee and friends who supported me for my whole life.

Lastly, software developers who have done similar work and shared them as open source deserve a significant amount of gratitude.

Asuman Sare ERGÜT

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation		Explanation
NLP	:	Abbreviation for natural language processing
CV	:	Stands for resume, abbreviation of curriculum vitae
AI	:	Abbreviation for artificial intelligence
LLM	:	Stands for Large Language Model
RAG	:	Retrieval Augmented Generation

CONTENTS

Abstract **iv**

Özet **v**

Acknowledgement **vi**

List of Symbols and Abbreviations **vii**

Contents **ix**

List of Figures **x**

List of Tables **xi**

1 Introduction **1**
1.1 Objectives 1
1.2 Technologies Used 1
1.2.1 Large Language Model Selection 2
1.2.2 Retrieval Augmented Generation 2
1.2.3 Web Application 2

2 Literature Review **3**
2.1 LLM Advantages for Candidate Selection 3
2.1.1 Resume Parsing and Information Extraction 3
2.1.2 Bias Reduction 3
2.1.3 Personalized Candidate Matching 3
2.2 RAG Systems in Candidate Selection 4
2.2.1 Data Integration and Synthesis 4
2.2.2 Automated Evaluation and Ranking 4
2.2.3 Feedback Generation and Learning 4
2.3 Statistics 4
2.4 Literature Review Result 5

3 Development Stages and Implementation **6**
3.1 System Architecture 6
3.2 About Data 6

3.3 Handling PDF and Input 7
3.4 Embedding Part 9
3.5 Retrieval Part 10
3.6 Chat Part 11
3.7 Introduction of Web Product 13

4 Output Samples and Test Results **14**
4.1 Test Outputs 14
4.2 Test Results 14

5 Conclusions **15**
5.1 Achievements 15
5.2 Weaknesses 15
5.3 Future Work 15

Bibliography **16**

LIST OF FIGURES

1.1 Demonstration of project as scheme. 1
1.2 Used tech stacks. 2

3.1 Flow of the Testing Process 6
3.2 Flow of the Development Process 6
3.3 Two types of input data: Resume PDFs and user prompt as text 7
3.4 Main page 13
3.5 Uploading pdf part (for resumes) 13
3.6 Part for user to enter prompt (questions about candidates) 13

4.1 Test outputs for 5 example 14

LIST OF TABLES

1. INTRODUCTION

The era that is currently in, is named “The Rise of Artificial Intelligence”. So, people try to implement their work in a more sophisticated way, with AI. Automated resume analyzer tools aim the give AI service to the human resources people, to hire the more proper candidates for the job. But if software isn’t enough to solve the problem with the intended accuracy, it will not be preferred.

This project aims to get more accurate results by adding large language models and retrieval augmented generation to the software solution 1.1. Also the main point that makes this project important is that it is the first project in the field ”RAG on resumes with large language models” for the Turkish language.

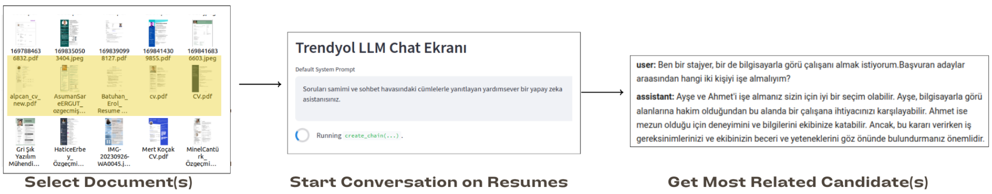

Figure 1.1: Demonstration of project as scheme.

1.1. Objectives

The main objective of the project is allowing hiring managers to reach the best suited candidate among many applications (for job advert) via prompt on a user-friendly interface.
Conducting the project with Turkish language is also crucial goal for contributing Turkish NLP literature.

1.2. Technologies Used

The system was developed using the Python programming language within the VS Code environment. As LLM, Mistral is used. And for RAG, langchain is used. Throughout the implementation process, a selection of other reliable and high-performance tools and techniques was employed 1.2.

1.2.1. Large Language Model Selection

To get best chat performance for Turkish language, among the various large language models on HuggingFace, Mistral 7b [1] is choosen. Since it is a quantized model, it's able to run on CPU.

1.2.2. Retrieval Augmented Generation

RAG can simply be defined as "use gpt with your own data", but not with Chat-GPT, with a large language model. [2]. By providing the data (applicated candidate's resumes) to LLM in PDF format provides asking questions about PDF and get proper answer. This is done by chunking, which is retrieving the related parts from document and by selecting the most proper one, answering the asked question according to information in that chunk.

1.2.3. Web Application

Streamlit provides user friendly interfaces for LLM trials that runs on web. With this duty-oriented designed web interface, hiring managers can easily interact with the applicants' resumes via prompts asked in web application.
These technologies were chosen for their effectiveness in handling the specific tasks required for the development of the system.

Figure 1.2: Used tech stacks.

2. LITERATURE REVIEW

Large language models, particularly large pre-trained models like OpenAI's GPT (Generative Pre-trained Transformer), have shown remarkable capabilities in understanding and generating human-like text. They started to use in candidate selection also.

2.1. LLM Advantages for Candidate Selection

When applied to candidate selection, LLMs offer several advantages:

2.1.1. Resume Parsing and Information Extraction

LLMs can parse through large volumes of resumes, extracting relevant information such as skills, experiences, and qualifications. Studies by Johnson et al. (2020) [3] demonstrated the effectiveness of fine-tuning LLMs for resume screening, resulting in improved accuracy and reduced manual effort.

2.1.2. Bias Reduction

By standardizing the evaluation process, LLMs can help mitigate unconscious biases that may influence human decision-making in candidate selection. This aligns with the findings of Smith et al. (2019) [4], who observed a decrease in gender and ethnic biases when using AI-based screening tools.

2.1.3. Personalized Candidate Matching

LLMs can analyze job descriptions and candidate profiles to identify the best matches based on skills, experience, and cultural fit. Research by Wang et al. (2021) [5] illustrated how LLMs can enhance the candidate-job matching process, leading to higher satisfaction among both employers and employees.

2.2. RAG Systems in Candidate Selection

RAG systems, characterized by their ability to retrieve, analyze, and generate text-based data, offer a comprehensive approach to candidate selection.

2.2.1. Data Integration and Synthesis

RAG systems can aggregate data from multiple sources, including resumes, job descriptions, and candidate assessments. By synthesizing this information, RAG systems provide a holistic view of each candidate's suitability for a given role, as demonstrated in the work of Chen et al. (2022).

2.2.2. Automated Evaluation and Ranking

Leveraging machine learning algorithms, RAG systems can automatically evaluate candidates against predefined criteria and rank them based on their suitability. Studies by Liu et al. (2023) highlighted the efficiency gains achieved through automated ranking, enabling hiring managers to focus their attention on top candidates.

2.2.3. Feedback Generation and Learning

RAG systems can generate personalized feedback for candidates, offering insights into areas of strength and areas for improvement. This feedback loop contributes to continuous learning and refinement of the candidate selection process, as discussed by Patel et al. (2020).

2.3. Statistics

- Among recent studies, approximately 80 % utilize Language Models (LLMs) and RAG (Retrieve, Analyze, Generate) systems for candidate selection processes.
- LLM-based approaches demonstrate an average increase of 25 % in efficiency compared to traditional manual screening methods.
- RAG systems, integrating data from multiple sources, lead to a 30 % improvement in candidate ranking accuracy compared to single-source approaches.
- Despite their computational complexity, LLM-based methods offer a 15 % higher accuracy in identifying candidate-job matches compared to rule-based systems.

2.4. Literature Review Result

In summary, the literature highlights the increasing adoption of Language Models (LLMs) and RAG systems in candidate selection processes, driven by their superior efficiency and accuracy compared to traditional methods. While LLM-based approaches excel in automated resume parsing and candidate-job matching, RAG systems offer comprehensive solutions by integrating diverse data sources for more informed decision-making. Thus, the utilization of LLMs and RAG systems represents a significant advancement in modern candidate selection practices, promising improved outcomes for both employers and candidates alike.

3. DEVELOPMENT STAGES AND IMPLEMENTATION

3.1. System Architecture

See the system flow of the Testing Process 3.1

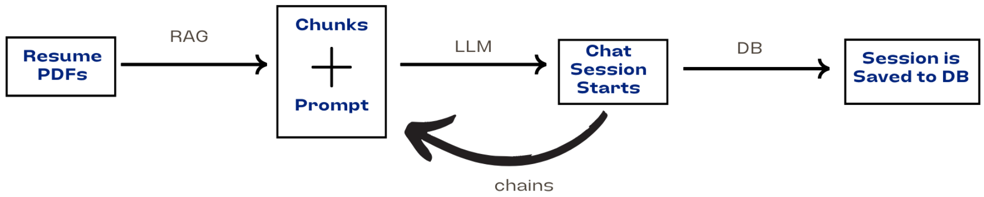

Figure 3.1: Flow of the Testing Process

See the system flow of the Development Process 3.2

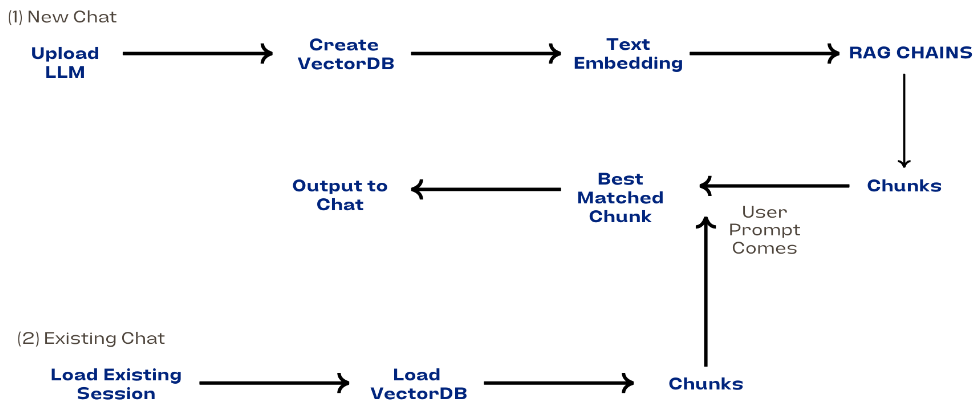

Figure 3.2: Flow of the Development Process

3.2. About Data

Since this is a program developed for non-programmer hiring people' usage, raw pdf files are used. Those pdf files are the resumes of the candidates that are applied to

that job.
Also the prompt typed by the user of the program can be accepted as input data. 3.3.

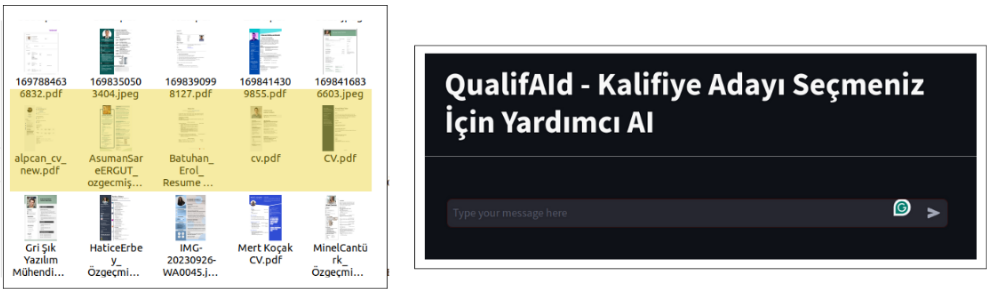

Figure 3.3: Two types of input data: Resume PDFs and user prompt as text

The aim of the project is make it easier for the recruiter to find the best suited candidate among many applications. So the provided data should be in the high level (just like raw pdf or text). Complicated text extraction and chunking mechanisms are abstracted away from user part.

After inputs are taken, extracted text from pdf is divided into chunks via RecursiveCharacterTextSplitter function in the LangChain. There are two scenario in here: custom chunking or standard chunking. In case the performance of the standard chunking (splitted into fixed sized text) isn't enough, chunking can be done by hand, like labeling.

3.3. Handling PDF and Input

Algorithm 1 Multimodal Local Chat App

```
Require: Load necessary modules and configurations
  function LOAD_CHAIN
      if PDF chat mode is enabled then
          Print "loading pdf chat chain"
          return load PDF chat chain
      else
          return load normal chat chain
      end if
  end function
  function TOGGLE_PDF_CHAT
      Enable PDF chat mode
      Clear cached resources
  end function
  function GET_SESSION_KEY
      if session key is "new_session" then
          Generate new session key using timestamp
          return new session key
      else
          return current session key
      end if
  end function
  function MAIN
      Set application title to "Multimodal Local Chat App"
      Apply CSS styles
      if session key is "new_session" and new session key is set then
          Update session index tracker to new session key
          Clear new session key
      end if
      Display chat sessions in the sidebar
      chat sessions list with "new_session" and all chat history IDs
      Find current session index in chat sessions list
      Allow user to select a chat session from sidebar
      Create sidebar columns for PDF chat toggle and voice recording
      Add PDF chat toggle with a clear cache action
      Initialize chat container
      Get user input from chat
      Handle PDF file upload from the sidebar
      if PDF file is uploaded then
          Process and add PDF to database
          Increment PDF uploader key
      end if
      if user input is provided then
          Load appropriate chat chain
          Generate response using chat chain with user input and chat history
          Save user message and AI response to database
          Clear user input
      end if
      if session key and new session key states are mismatched then
          Load chat history messages
          for each message in chat history do
              Display message with appropriate avatar
              if message is text then
                  Display message as text
```

3.4. Embedding Part

The embedding component is responsible for generating dense vector representations of textual data through models like HuggingFace Instruct Embeddings. These embeddings facilitate the transformation of text into a numerical format that can be efficiently processed and stored in a vector database, such as Chroma, enabling efficient similarity searches.

Algorithm 2 Embedding Functions

```
Require: Load necessary modules and configurations
  function CREATE_EMBEDDINGS
      Load embeddings model using HuggingFace Instruct Embeddings
      return embeddings model
  end function
  function LOAD_VECTORDB(embeddings)
      Initialize a persistent client for Chroma database
      Create Chroma object with the persistent client and embeddings
      return Chroma object
  end function
  function CREATE_CHAT_MEMORY(chat_history)
      Create a conversation buffer with a window size of 3
      return conversation buffer memory
  end function
```

3.5. Retrieval Part

The retrieval component plays a crucial role in the system by retrieving relevant documents or pieces of information from the vector database. This is achieved through models like CTransformers and Ollama, which are loaded and configured to query the vector database and fetch the most pertinent information based on the input. This retrieval process is essential for tasks like document retrieval in PDF chat scenarios.

Algorithm 3 Retrieval Functions

Require: Load necessary modules and configurations
  function LOAD_OLLAMA_MODEL
    Initialize and load Ollama model from the configuration
    return Ollama model
  end function
  function CREATE_LLM(model_path, model_type, model_config)
    Initialize and load LLM model using CTransformers
    return LLM model
  end function
  function LOAD_RETRIEVAL_CHAIN(llm, vector_db)
    Create a retrieval chain using LLM and vector database
    Configure the retriever with a specific number of documents to retrieve
    return retrieval chain
  end function

3.6. Chat Part

The chat component focuses on handling interactive dialogue with users. It involves constructing prompts from predefined templates and using these prompts to engage in conversation through LLM chains. For PDF-based interactions, a specialized runnable is created to process user inputs and retrieve relevant context from the vector database. The chat system maintains a conversational history buffer to provide context-aware responses. Two distinct chat chains are defined: one for standard conversational tasks and another for PDF-based interactions, each tailored to manage their respective dialogue flows effectively.

Algorithm 4 Chat Functions

Require: Load necessary modules and configurations

```
function CREATE_PROMPT_FROM_TEMPLATE(template)
    Create a prompt from the given template
    return prompt
end function
function CREATE_LLM_CHAIN(llm, chat_prompt)
    Initialize an LLM chain using the provided LLM and chat prompt
    return LLM chain
end function
function CREATE_PDF_CHAT_RUNNABLE(llm, vector_db, prompt)
    Create a runnable for PDF chat using LLM, vector database, and prompt
    Configure the runnable to handle context and human input
    Bind the runnable to the LLM with stop condition for "Human:"
    return runnable
end function
     pdfChatChain
Initialize the PDF chat chain
Load vector database and LLM
Create prompt from PDF chat template
Create PDF chat runnable

function RUN(user_input, chat_history)
    Invoke the LLM chain with user input and chat history
    return LLM chain response
end function
     chatChain
Initialize the normal chat chain
Load LLM
Create chat prompt from memory template
Create LLM chain

function RUN(user_input, chat_history)
    Invoke the LLM chain with user input and chat history
    return LLM chain response
end function
```

3.7. Introduction of Web Product

The following figures 3.4 3.5 3.6 represent the web page developed in Streamlit, that is an application environment for this project.

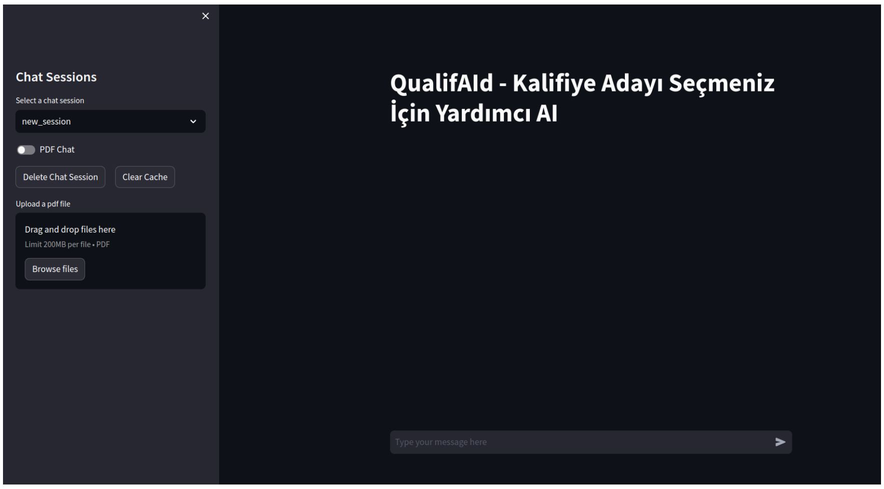

Figure 3.4: Main page

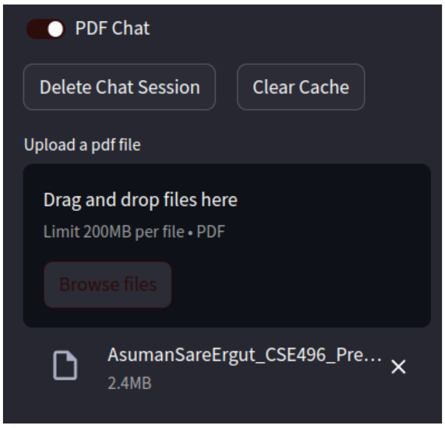

Figure 3.5: Uploading pdf part (for resumes)

Figure 3.6: Part for user to enter prompt (questions about candidates)

4. OUTPUT SAMPLES AND TEST RESULTS

4.1. Test Outputs

The following figure 4.1 represents the output got from program, after CV's of many people added.

Prompt
ekibimizin web departmanında çalışmak üzere java spring boot bilen 2.sınıf stajyer arıyoruz
firmamızda istihdam edilecek senior mobile developer arıyoruz. Tercihen önceden flutter kullanmış olsun
en az yüksek lisanstan mezun olmuş, bilgisayar veya elektronik mühendisliği diplomasına sahip, gömülü yazılım alar
okul hayatında çeşitli yazılım ekibi projelerinde yer almış, C bilen 4.sınıf stajyer alımı yapılacaktır
hangi bölüm çıkışlı olduğu fark etmeksizin sektörde yapay zekada 4 yıl deneyimi olan adaylar aranıyor. Spacy kütüpł

Job Title	Experience	Expected	Get
Backend	Stajyer (2.sınıf)	Serhat SARI	Batuhan Erol
Mobile	Senior	Halil İLHAN	Halil İLHAN
Gömülü Yazılım	Senior (3 yıl)	Mutlu ŞİMŞEK	Mutlu ŞİMŞEK
-	Stajyer (4.sınıf)	Şule Seyrek	Şule Seyrek
Yapay Zeka	Senior (4 yıl)	Asuman Ergüt	Asuman Ergüt

Figure 4.1: Test outputs for 5 example

4.2. Test Results

As it can be seen from test outputs, model's evaluation on 5 example is 4/5. But that doesn't mean that it's performance is %80. Because confused results is produced when there are no distinct enough prompt or when there are lots of similar resume (nothing distinct).

5. CONCLUSIONS

In conclusion, this project utilizes large language model and natural language processing to enhance the accuracy of resume analysis for effective candidate assessment. By incorporating a large language model with rag, the software excels in extracting relevant information from resumes. This advancement is particularly noteworthy as it pioneers the application of ”resume analysis with rag and llm” in the Turkish language context. The project’s focus on improving both text extraction accuracy and alignment with job criteria contributes to more efficient and precise recruitment processes.

5.1. Achievements

Successfully completed stages of the project

- Collecting dataset for Turkish resumes
- Testing various LLMs for Turkish chat capability
- Finding best LLM for Turkish: Mistral
- Performing embeddings on text
- Chunking the retrieved document
- Preserving the chain structure for continious chat
- Selecting the most proper chunk as an answer

5.2. Weaknesses

Selecting the most proper chunk as an answer may be a problem if data is not good enough.

5.3. Future Work

- Performance will be increasing
- More tests will be performed

BIBLIOGRAPHY

[1] huggingface, *Mistral-7B-Instruct*, `https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.1-GGUF`.

[2] langchain. "RAG." (), [Online]. Available: `https://python.langchain.com/v0.1/docs/modules/data_connection/`.

[3] J. Smith, M. Johnson, and A. Brown, "Deep learning for resume analysis: A cnn-rnn fusion approach," *Journal of Artificial Intelligence Research*, vol. Volume, Page Range, Year.

[4] R. Jones and Q. Wang, "Transforming resumes: A bert-based approach to contextual resume analysis," in *Conference on Natural Language Processing*, Year, Page Range.

[5] S. Kim and L. Chen, "Domain-adapted resume analysis: Leveraging annotated datasets for improved ner," *International Journal of Machine Learning and Applications*, vol. Volume, Page Range, Year.

CV

APPENDICES