

Resume Enhancement and Analysis App

INFO 7390, Fall 2023

Presented by

Shruti Tambe 002762916

Sanjay Bhaskar Kashyap 002780659

INTRODUCTION

Purpose: This project aims to develop a web application that assists individuals in enhancing their resumes. It leverages advanced Natural Language Processing (NLP) techniques to analyze resumes, suggest improvements, and provide answers to user queries based on the resume content.

Target Audience: The primary audience for this app includes job seekers, career advisors, and anyone interested in refining their resume to better match job descriptions.

TECHNOLOGY USED

- **Python** - Chosen for its strong support in data science and NLP libraries
- **Streamlit app** - An efficient framework for building interactive web apps entirely in Python
- **Spacy** - Utilized for efficient and accurate NLP tasks, particularly in keyword extraction and text analysis
- **PyPDF2** - To handle PDF file reading and text extraction
- **Transformers (BERT)** - Employed for its state-of-the-art performance in NLP tasks, particularly in question answering.

... But first let us have a look
at our previous
Resume Question Answering Model

Previous Model: Resume Question Answering Model

- **User Uploads Resume:** Users start by uploading a resume.
- **Question Input:** After uploading the resume, users can input questions related to the resume content.
- **Model Processing:** The BERT-based model processes the resume and the questions, leveraging its contextual understanding to provide accurate responses.
- **Answer Display:** The model returns answers to the user's questions, displayed in the Streamlit application.
- **Feedback Collection:** Users are encouraged to provide feedback on the quality of the answers. They can indicate whether the answers were satisfactory or not.

Methodology - BERT Language Model

```
# Initialize session state to store the log of QA pairs and satisfaction responses
if 'qa_log' not in st.session_state:
    st.session_state.qa_log = []

def extract_text_from_pdf(pdf_file):
    pdf_reader = PyPDF2.PdfReader(BytesIO(pdf_file.read()))
    text = ""
    for page in pdf_reader.pages:
        text += page.extract_text()
    return text

def answer_question(question, context, model, tokenizer):
    inputs = tokenizer.encode_plus(
        question,
        context,
        add_special_tokens=True,
        return_tensors="pt",
        truncation="only_second",
        max_length=512,
    )
    outputs = model(**inputs, return_dict=True)
    answer_start_scores = outputs.start_logits
    answer_end_scores = outputs.end_logits
    answer_start = torch.argmax(answer_start_scores)
    answer_end = torch.argmax(answer_end_scores) + 1
    input_ids = inputs["input_ids"].tolist()[0]
    answer = tokenizer.convert_tokens_to_string(
        tokenizer.convert_ids_to_tokens(input_ids[answer_start:answer_end])
    )
    return answer

st.title("Resume Question Answering")
```

```
uploaded_file = st.file_uploader("Upload your resume (PDF format only)", type=["pdf"])

if uploaded_file is not None:
    resume_text = extract_text_from_pdf(uploaded_file)
    st.write("Resume Text:")
    st.write(resume_text)

    user_question = st.text_input("Ask a question based on your resume:")

    if user_question:
        model = BertForQuestionAnswering.from_pretrained("bert-large-uncased-whole-word-masking-finetuned-squad")
        tokenizer = BertTokenizer.from_pretrained("bert-large-uncased-whole-word-masking-finetuned-squad")

        answer = answer_question(user_question, resume_text, model, tokenizer)
        st.write("Answer:")
        st.write(answer)

        # Ask for user feedback on satisfaction
        satisfaction = st.radio('Are you satisfied with the answer?', ('Yes', 'No'), key='satisfaction')

        # Log the interaction
        st.session_state.qa_log.append({
            'Question': user_question,
            'Answer': answer,
            'Satisfaction': satisfaction
        })

        # Display the log in a table format
        st.write("Interaction Log:")
        log_df = pd.DataFrame(st.session_state.qa_log)
        st.dataframe(log_df)
```

Results for BERT language model

Resume Question Answering

Upload your resume (PDF format only)

Drag and drop file here
Limit 200MB per file • PDF

Browse files

Resume_Sanjay_Bhaskar_Kashyap.pdf 153.7KB

×

Ask a question based on your resume:

What is the phone number of the candidate?

Answer:

945 - 244 - 7079

Are you satisfied with the answer?

☒ Yes

☐ No

| | Question | Answer | Satisfaction |
|---|--|------------------|--------------|
| 0 | What is the name of the candidate? | sanjay bhaskar | Yes |
| 1 | What is the phone number of the car | 945 - 244 - 7079 | Yes |
| 2 | What is the education details of the c | master of scienc | Yes |
| 3 | What are the skills of the candidate? | programming lai | Yes |
| 4 | Does the candidate have data scienc | data science me | Yes |
| 5 | Does the candidate have data scienc | data science me | No |
| 6 | What is the candidate's professional | master of scienc | No |
| 7 | What is the name of the candidate? | sanjay bhaskar | Yes |
| 8 | What is the name of the candidate?\ | sanjay bhaskar | Yes |
| 9 | What is the phone number of the car | 945 - 244 - 7079 | Yes |

Results for BERT language model

Ask a question based on your resume:

What is the name of the candidate? \ Press Enter to apply

Answer:

sanjay bhaskar

Are you satisfied with the answer?

☒ Yes

☐ No

Accurate results

•President of Theater club, CMRIT

Ask a question based on your resume:

Did the candidate work as an intern?

Answer:

work experience analytics specialist intern , havas media , boston june 2023 – august 2023

Are you satisfied with the answer?

☒ Yes

☐ No

Fairly Accurate results

Other Methods we tried: GPT-2 Model

```
# Load pre-trained GPT-2 model and tokenizer
model_name = 'gpt2-medium'
model = GPT2LMHeadModel.from_pretrained(model_name)
tokenizer = GPT2Tokenizer.from_pretrained(model_name)

# Function to generate answers based on the resume text
def generate_answer(resume_text, question):
    # Extract relevant parts of the resume text
    relevant_text = extract_relevant_text(resume_text, question)

    text = relevant_text + " " + question
    input_ids = tokenizer.encode(text, return_tensors='pt', truncation=True, max_length=1024)

    # Set attention mask and pad token id
    attention_mask = torch.tensor([[1] * len(input_ids[0])])
    pad_token_id = tokenizer.eos_token_id

    output = model.generate(input_ids, attention_mask=attention_mask, pad_token_id=pad_token_id, max_length=1024, num_return_sequences=1)
    answer = tokenizer.decode(output[0], skip_special_tokens=True)
    return answer

# Function to extract relevant parts of the resume text
def extract_relevant_text(resume_text, question):
    # Extract the parts of the resume text that are relevant to the question
    # Here, I'm using a simple string matching approach
    relevant_text = ""
    for sentence in resume_text.split('.'):
        if any(word in sentence for word in question.split()):
            relevant_text += sentence + '.'

    return relevant_text

# Streamlit app
st.title('Resume Question Answering App')
```

```
# Streamlit app
st.title('Resume Question Answering App')

# Resume PDF upload
uploaded_file = st.file_uploader('Upload your resume (PDF format):', type='pdf')

# Extract text from the uploaded PDF
if uploaded_file is not None:
    pdf_reader = PyPDF2.PdfReader(BytesIO(uploaded_file.read()))
    resume_text = ''
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        resume_text += page.extract_text()
    st.write('Resume Text:')
    st.write(resume_text)

# User question input
user_question = st.text_input('Ask a question based on your resume:')

# Generate answer based on user question and resume text
if user_question:
    answer = generate_answer(resume_text, user_question)
    st.write('Answer:')
    st.write(answer)
```

Other Methods we tried : TF-IDF Vectorizer

```
def extract_text_from_pdf(pdf_file):
    pdf_reader = PyPDF2.PdfReader(BytesIO(pdf_file.read()))
    text = ""
    for page in pdf_reader.pages:
        text += page.extract_text()
    return text

# Load job skills data
data = pd.read_csv("C:/Users/19452/Downloads/jobss.csv")

# Preprocess data
data = data.dropna(subset=['Key Skills'])
data['Key Skills'] = data['Key Skills'].str.replace('|', ' ')

# Train a TF-IDF vectorizer on the job skills data
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['Key Skills'])

# Define a function to answer questions
def answer_question(question, data, vectorizer, X):
    query_vec = vectorizer.transform([question])
    cosine_similarities = cosine_similarity(query_vec, X).flatten()
    most_similar_idx = cosine_similarities.argmax()
    answer = data.iloc[most_similar_idx]['Job Title']
    return answer
```

```
# Streamlit app
st.title("Resume Question Answering")

uploaded_file = st.file_uploader("Upload your resume (PDF format only)", type=["pdf"])

if uploaded_file is not None:
    resume_text = extract_text_from_pdf(uploaded_file)
    st.write("Resume Text:")
    st.write(resume_text)

    user_question = st.text_input("Ask a question based on your resume:")

    if user_question:
        answer = answer_question(user_question, data, vectorizer, X)
        st.write("Answer:")
        st.write(answer)
```

Results for TF-IDF Vectorizer

Ask a question based on your resume:

what is the ideal role for this candidate?

Answer:

Consultant

Fairly accurate result

Ask a question based on your resume:

what is the phone number of this candidate?

Answer:

UI Developer (Frontend Developer)

Highly inaccurate
result for the given
input

Now back to enhancements for our Final Project

APPLICATION OVERVIEW

The application provides various features:

- **Text Extraction from PDF Resumes:** Extracts user-uploaded resumes in PDF format.
- **Question Answering:** Uses a BERT model to answer questions based on the resume content.
- **Keyword Extraction:** Extracts keywords from resumes and job descriptions to suggest improvements.
- **Resume and Job Description Analysis:** Compares the two documents for matching keywords.

NLP Techniques

BERT for Question Answering

- The BERT model, specifically the *bert-large-uncased-whole-word-masking-finetuned-squad*, is used for its excellence in understanding the context of a word in a sentence. The model, pre-trained on a vast corpus and fine-tuned on question-answering tasks, can comprehend and provide precise answers to user queries based on their resume.

Keyword Extraction and Resume Analysis

- The *extract_keywords_for_sections* function uses Spacy to identify keywords in both the resume and the job description. The app then suggests improvements and identifies potential project ideas based on these keywords.


Spacy for Keyword Extraction and Pattern Matching

- Spacy is used for tokenizing the resume and job description texts, tagging each token with its part of speech. It then uses pattern matching to identify skills, technologies, and project ideas, thereby enabling effective keyword extraction. This extraction is pivotal in analyzing and suggesting enhancements for the resume.

User Interface and Interaction


- Developed using Streamlit, the app provides a clean and interactive interface.
- Users can upload their resumes, input job descriptions, ask questions, and receive tailored suggestions.
- The interface is designed to be user-friendly, allowing seamless navigation through various features.

Results



Resume Enhancement and Analysis App

Upload your resume (PDF format):

 Drag and drop file here
Limit 200MB per file • PDF

Browse files

Input the job description here for resume improvement suggestions:

Ask a question based on your resume:

What is the candidate's name ?

Answer:

sanjay bhaskar

Results

Input the job description here for resume improvement suggestions:

Data Engineer with SNOWFLAKE and have good data analytics experience

- They would love to hire a candidate that can grow into a lead and help to mentor the data analysts on the team

Technical Skill Improvement Suggestions:

Consider highlighting your experience or skills related to 'models'.

Consider highlighting your experience or skills related to 'Google Analytics'.

Consider highlighting your experience or skills related to 'designing'.

Consider highlighting your experience or skills related to 'communication skills'.

Consider highlighting your experience or skills related to 'Google'.

Notable Project Ideas:

Consider a project involving 'using Snowflake'.

Consider a project involving 'building data models'.

Consider a project involving 'building data'.

Match Commentary:

Your resume matches the following keywords from the job description: team, Data, Snowflake, experience, Analytics, engineering, analytics, Experience, SQL, data, tools, analysts, Tableau

Interaction Log:

Are you satisfied with the answer to: "What is the candidate's name ?"?

- ☒ Yes
- ☐ No

| | Question | Answer | Satis |
|---|----------------------------------|---|-------|
| 0 | What are the candidate's skills? | programming languages : python , r , java , sql , c , c + + , html , matlab | Yes |
| 1 | What is the candidate's name ? | sanjay bhaskar | Yes |
| 2 | What is the candidate's name ? | sanjay bhaskar | Yes |

Challenges and Solutions

- One of the primary challenges was ensuring the accuracy of NLP tasks.
- This was addressed by carefully selecting and fine-tuning the BERT and Spacy models.
- Another challenge involved creating an intuitive user interface, which was resolved using Streamlit's straightforward framework.

Future Enhancements

Future plans include integrating more advanced NLP models for broader language support, enhancing the app's scalability, and incorporating real-time resume editing features.

Conclusion

- This project successfully demonstrates the use of cutting-edge NLP techniques in a practical application.
- It offers valuable assistance in resume enhancement, catering to the needs of job seekers and career professionals alike.