Resume Analysis Tool

DATS 6312: Natural Language Processing
Fall 2023

Instructor: Dr. Amir Jafari

Final Project Report

Group 1

Jadhav, Shubham
G30570862
Department of Computer Science
School of Engineering
George Washington University

Ramagiri, Jahnavi
G46015075
Department of Computer Science
School of Engineering
George Washington University

GitHub Repository: https://github.com/JahnaviRamagiri/Final-Project-Group1

December 11, 2023

Abstract

This project aimed to improve the efficiency of hiring by automatically classifying resumes into the most suitable job roles. We compared various models, from simple tools like Naive Bayes to advanced transformer-based models like BERT using PyTorch. One unique aspect of our work was a resume-to-job description comparison. This "similarity score" helps measure how well a resume matches a specific job posting. Additionally, we developed a system to automatically generate informative summaries of resumes.

By comparing these different models, we gained valuable insights into how to best analyze resumes. Our work not only helps companies find the right candidates for their jobs but also provides valuable information for researchers in the field of natural language processing.

Keywords—Resume, Train, Test, Loss, Classification, Transformer

Contents

List of Figures

List of Tables

1 Introduction

The primary motivation for this project is to address the challenges candidates face when applying for job roles. Every candidate goes through the time-consuming process of shortlisting job postings based on their resumes manually. The difficulty in accurately understanding if they or their resume is well suited for the job requirement is alarming.

The Resume Analysis tool, aims to simplify and expedite the job application process, benefiting both candidates and also recruiters (future extension). Through the use of advanced NLP techniques, it seeks to provide more accurate and efficient resume analysis and job role matching. This aims to streamline this process by leveraging NLP techniques to achieve the following objectives:

1. **Resume - Job Description Similarity Score**: Calculate the similarity score between a candidate's resume and a job description to assess how well the resume matches the job requirements.
2. **Multi-class Resume Classification**: Categorize resumes into predefined classes or job roles, making it easier for candidates to identify suitable job roles for their resume.
3. **Resume Analysis**: Provide comprehensive analysis and insights into a candidate's resume, including key skills, qualifications, and experience.

2 Dataset

Source Link: https://github.com/florex/resume_corpus/tree/master

The multi-labeled dataset of resumes labeled with occupations. The resume files have the extension ".txt" and the corresponding labels are in a file with the extension ".lab".

This dataset contains 3 files :

1. resumes_corpus.zip: This file contains a set of resumes files with the extension ".txt" with the corresponding list of labels in a file with the extension ".lab"
2. resumes_sample.zip : This file represents the data set of resumes in a single text file. Each line of the file contains information about a text resume. Each line has 3 fields separated by ":::". The first field is the reference id of the resume; the second field is the list of occupations separated by ";" ; and the third field is the text resume.
3. normalized_classes : This file contains the associations between the occupations as written by the experts and their corresponding normalized form.

3 Experimental Setup

3.1 Architecture

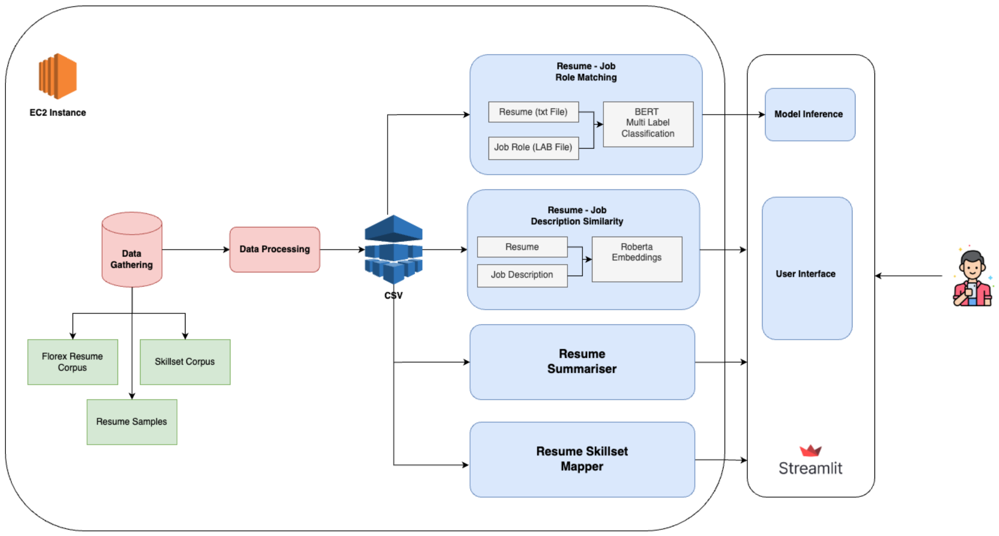


Figure 1: Model Architecture of proposed system

The model architecture for the Resume Analyzer project is depicted in the provided diagram, encompassing various integral components. The central focal point is the **Resume - Job Role Matching** module, utilizing BERT for multilabel classification. In the initial stage, data is gathered through **Data Gathering** from key datasets: Florex Resume Corpus, Skillset Corpus, and Resume Samples. The collected data then undergoes processing in the **Data Processing** module before being forwarded to subsequent modules. A specialized **Resume Skill Set Mapper** is employed for extracting skill sets from the given resumes. The **Resume Summarizer'** constitutes a pretrained model that delivers an overarching summary of the resume. Furthermore, a **Resume - Job Description Similarity** module assesses the similarity and suitability of a resume to a given job description. An EC2 Instance is integrated for computational processing or hosting purposes. Finally, the **Model Inference** component marks the phase where the model makes insightful predictions and inferences.

3.2 Data Preprocessing

Data preprocessing is a crucial step in the data analysis and machine learning pipeline. It involves cleaning and transforming raw data into a format that is suitable for analysis or for training machine learning models. The quality of the data and the features extracted from it significantly impact the performance of any data-driven project.

3.2.1 Data Gathering

The initial phase of the project starts with accumulating all the available data. From section 2 we can see that we have 2 zip files and one normalized classes text file. The objective here is to create dataset in pandas which are easier to manipulate for various tasks. The resume_corpus file has two type of files i.e., txt file with resume and lbl file with job roles for the corresponding resume. We iterate a loop over all the files and create a dataset with two columns. First column has resume text and second column with job roles seperated with comma. The resume_samples file is a single text file with all the resume and corresponding skillset in each sentence for each resume. We import this file in python and loop over sentence and extract resumes and skillset which are seperated by ":::". All the files are stored as Uncleaned dataset.

3.2.2 Data Cleaning

Once the data is curated and placed in the Uncleaned Folder, the next step is to clean it and process it for the next models. Data Cleaning is mainly done in on Resumes and SkillSet.

Resume: The clean_resume_text method processes individual resume texts by:

1. Removing HTML tags using BeautifulSoup.
2. Eliminating special characters and punctuation with regular expressions.
3. Lowercasing all text for consistency.
4. Tokenizing the text into individual words.
5. Lemmatizing tokens for standardization.
6. Removing duplicate neighboring tokens.
7. Eliminating common English stopwords.
8. Concatenating cleaned tokens into a single string for further analysis.

Skillset: The clean_skillset method processes individual skillsets by:

1. Checking for NaN values and returning an empty string if the skillset is invalid.
2. Lowercasing the entire skillset for uniformity.
3. Removing content within parentheses using a regular expression.
4. Tokenizing the skillset by replacing '/' with ';' and splitting into a list of individual skills.
5. Removing duplicate skills by converting the list to a set and then back to a list.
6. Sorting the list of skills alphabetically.

7. Presenting the final cleaned skillset as a list of individual skills.

Normalised Classes: The clean_skills_norm method processes individual skillsets by:

1. Checking for NaN values and returning an empty string if the skillset is invalid.
2. Lowercasing the entire skillset for uniformity.
3. Removing content within parentheses using a regular expression.
4. Tokenizing the skillset by replacing '/' with ',' and splitting into a list of individual skills.
5. Removing duplicate skills by converting the list to a set and then back to a list.
6. Sorting the list of skills alphabetically.
7. Presenting the final cleaned skillset as a list of individual skills.

4 Skill Set matching from Resume

This modules extracts relevant skills from the Resume. The read_skillset function reads a CSV file containing normalized skillset data, creating a cleaned and stripped-down list of skills. This list, referred to as skill_list, becomes a key reference for skill matching. The subsequent functions, extract_skills_from_resume and identify_skillsets, utilize this reference to efficiently identify and extract relevant skills from given resume texts. The former scans the resume text for matches with each skill in the list, while the latter orchestrates the process, returning a set of unique skills identified in a particular resume.

Additionally, the script incorporates the @st.cache_data decorator to cache the results of the skill extraction functions. This caching mechanism enhances performance by avoiding redundant computations, ensuring that if the same resume text is encountered again, the previously identified skills are retrieved from the cache instead of reprocessing. Overall, these functions collectively contribute to a streamlined and optimized workflow for extracting skill sets from resumes, offering efficiency in the context of job application and matching.

5 Resume Summarization

In our project, we implemented a text summarization approach utilizing the **Hugging Face Transformers** library, specifically employing the "summarization" pipeline. The initial step involved extracting textual information from the resumes, followed by a basic data cleaning process to enhance the quality of the input data. Leveraging the transformer-based **BART (Bidirectional and Auto-Regressive Transformers)** model, the summarizer was applied to generate coherent and relevant summaries [1]. The integration of BART ensured the effectiveness of our summarization task, as it is specifically tailored for sequence-to-sequence tasks, such as summarization and text generation. The flexibility of the Hugging

Face pipeline allowed us to fine-tune summarization parameters, such as **max_length** to optimize the summarization results [2].

6 Resume-Job Description Similarity

We implemented two distinct approaches to enhance the analysis of resumes in relation to job descriptions. The first method utilized TF-IDF (Term Frequency-Inverse Document Frequency) Cosine Similarity, a traditional yet effective technique for representing document content and measuring the semantic similarity between documents [3, 4].

In addition to TF-IDF, we incorporated state-of-the-art transformer-based models, namely BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (Robustly optimized BERT approach) [5, 6]. For both the resume and job description, we tokenized the text and computed embeddings using BERT and RoBERTa Tokenizer. To obtain document-level embeddings, we employed mean pooling, providing a simplified representation of the texts. Lastly, we calculated the cosine similarity between the embeddings generated by BERT and RoBERTa between resume and job description.

The cosine similarity scores obtained from BERT and RoBERTa embeddings offer a more advanced and context-aware measure of similarity compared to the TF-IDF method.

$$\cos(\theta) = \frac{Resume \cdot JobDescription}{\|Resume\|_2 \, \|JobDescription\|_2}$$

7 Multi Label Classification

The primary goal of the resume analysis tool is to perform multilabel classification of resumes into suitable job roles. Given that a single resume can align with multiple job roles, this task falls under the category of multilabel classification. The dataset, resume_corpus, contains a mapping between .txt and .lab files, where the former holds the resume text, and the latter specifies the associated job roles. To kickstart this module, we opt for Multinomial Naive Bayes (MultinomialNB) as our foundational model. As a next step, we explore the implementation of a BERT Base model, adapting classifier heads to conduct experiments and identify the model configuration that yields the most promising results. This approach allows us to evaluate and compare the performance of both models, ensuring a comprehensive understanding of their effectiveness in addressing the multilabel classification challenge in the context of resume analysis.

7.1 Algorithms

The primary aim of the project to develop a multi-label classifier tasked with categorizing resumes into multiple job roles. To achieve this gaol, we will be experimenting with several alogorithms

7.1.1 Multinomial Naive Bayes Classifier

The Multinomial Naive Bayes (MultinomialNB) classifier is a powerful algorithm known for its simplicity and effectiveness in text classification tasks [7]. It is a probabilistic algorithm based on Bayes' theorem, with an assumption of independence between features.

To establish a baseline for comparison with state-of-the-art models, we have chosen to implement the Multinomial Naive Bayes (MultinomialNB) classifier. This classifier is renowned for its simplicity and effectiveness, making it a formidable candidate for text classification tasks, especially when dealing with discrete data such as word counts in documents [7]. The MultinomialNB classifier operates on the principles of Bayes' theorem, leveraging a probabilistic approach with an assumption of feature independence.

By employing the MultinomialNB classifier, we aim to assess its performance as a foundational model in the complex task of classifying resumes into multiple job roles. This classifier's simplicity and reliance on probabilistic reasoning make it an excellent candidate for comparison against more advanced models. The intention is to use the MultinomialNB classifier as a baseline to gauge the efficacy of state-of-the-art models, thus providing a valuable reference point in evaluating the advancements and nuances introduced by more sophisticated classifiers in the realm of multi-label resume classification.

$$P(A|B) = P(A) * \frac{P(B|A)}{P(B)}$$

7.1.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) has revolutionized natural language processing, particularly in sequence classification tasks. Developed by Google, BERT employs a transformer architecture to comprehensively capture contextual relationships in both directions within a sequence. Pre-trained on extensive corpora, BERT's bidirectional understanding allows it to create context-aware word representations. In the realm of sequence classification, such as resume analysis, BERT excels at discerning nuanced contextual meanings, making it a powerful tool for tasks like sentiment analysis and document categorization.

7.1.3 BERT+Linear

In the initial model architecture, BERT is coupled with a linear classifier for multiclass classification. This configuration represents a straightforward approach, leveraging the contextualized embeddings from BERT and connecting them to a linear layer that outputs class probabilities.

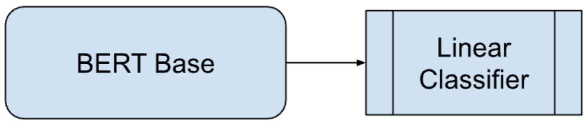


Figure 2: BERT with Linear head

In the BERT + Linear Classifier configuration, the BERT model is employed to generate contextualized word embeddings for each token in the input sequence. These embeddings encapsulate the semantic meaning of words in the context of the entire document. The resulting embeddings are then flattened or pooled to create a fixed-size representation of the entire sequence. This representation is then passed through a linear layer, which acts as a classifier. The linear layer maps the BERT embeddings to the number of output classes, producing a probability distribution over these classes. During training, the model learns the optimal weights for the linear layer by minimizing the Binary cross-entropy logits loss between predicted probabilities and actual labels.

7.1.4 BERT+LSTM

In the BERT + LSTM architecture, the BERT embeddings are further enriched with the sequential understanding capabilities of a Long Short-Term Memory (LSTM) layer. Unlike traditional neural networks, LSTMs maintain a memory cell that can capture dependencies over varying distances in the input sequence. This proves advantageous when dealing with tasks where the order of words or tokens is crucial for understanding context, such as in natural language processing. The LSTM layer processes the BERT embeddings in a sequential manner and provides a contextualized representation. This representation is then fed into a linear classifier, similar to the BERT + Linear approach, for the final classification.

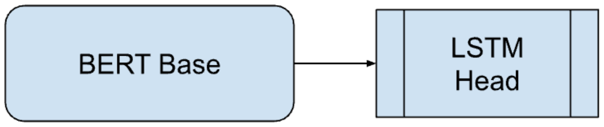


Figure 3: BERT with LSTM head

This model employs BERT embeddings and an LSTM layer for multilabel classification of resume data. The code begins by loading and preprocessing the dataset, utilizing the BertTokenizer to convert text into BERT-compatible input. The custom model, BertLSTMModel, integrates BERT with an LSTM layer for sequential feature extraction. The model is trained and evaluated using Adam optimization and BCEWithLogitsLoss. Training progress is tracked with tqdm, and evaluation metrics such as precision, recall, and F1 score are computed. The trained model parameters are saved, along with the corresponding labels for multilabel classification tasks. This implementation provides a flexible approach for leveraging contextual embeddings and sequential information in resume data classification.

7.1.5 BERT+CNN

The BERT + CNN architecture introduces Convolutional Neural Network (CNN) layers on top of the BERT embeddings. CNNs are adept at capturing local patterns and features within the data. In this configuration, 1D convolutional layers operate on the BERT embeddings, extracting relevant features by convolving over the token sequence. Pooling layers are applied to condense the spatial dimensions of the feature maps, and the resulting features are concatenated before being passed through a linear classifier. This approach is particularly useful when capturing specific patterns or structural information in the data is crucial for effective classification.

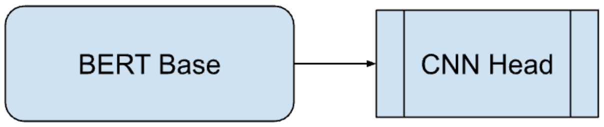


Figure 4: BERT with CNN head

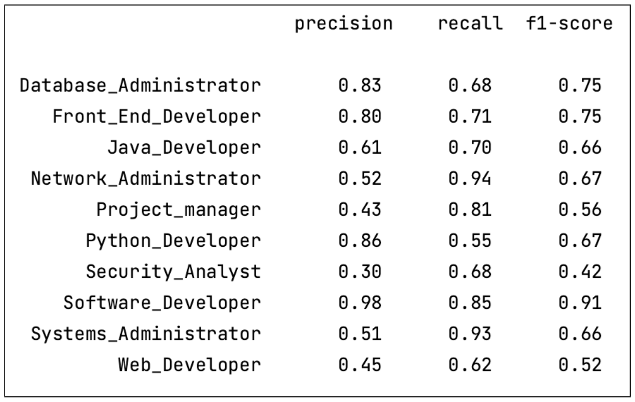
This model implements BERT Base with a CNN head for multilabel classification of resume data. The code starts by loading and preprocessing the dataset using the BertTokenizer. The custom model, BertCNNModel, integrates BERT with a 1D convolutional neural network (CNN) layer for feature extraction. The model is trained and evaluated using Adam optimization and BCEWithLogitsLoss. The training progress is monitored using tqdm, and the model's performance is evaluated in terms of precision, recall, and F1 score. The trained model parameters are saved along with the corresponding labels for multilabel classification tasks.

7.2 Model Optimization

1. **Learning Rate**: We experimented with a range from 0.001 to 2e-5.
2. **Batch Size**: We experimented with a range from 16 to 32 to identify the optimal balance for our model's memory constraints and learning stability.
3. **Epochs**: We restricted the number of epochs to a range of 3 to 10 to balance between underfitting and overfitting.
4. **Max Length**: The sequence length varied within the range of 64 to 256 tokens, enabling effective management of the context window.

7.3 Results

7.3.1 Multinomial Naive Bayes Classifier



```
                          precision    recall  f1-score


Database_Administrator         0.83      0.68      0.75
    Front_End_Developer        0.80      0.71      0.75
         Java_Developer        0.61      0.70      0.66
  Network_Administrator        0.52      0.94      0.67
        Project_manager        0.43      0.81      0.56
       Python_Developer        0.86      0.55      0.67
       Security_Analyst        0.30      0.68      0.42
     Software_Developer        0.98      0.85      0.91
  Systems_Administrator        0.51      0.93      0.66
          Web_Developer        0.45      0.62      0.52
```

Figure 5: Classification Report from Multinomial Naive Bayes Classifier

We can observe that the f1-score across all classes is approximately the same and in the range of 0.65 to 0.75 expect for Security Analyst, Project Manager, and Web Developer. The conclusive performance mertrics for the Multinomial Naive Bayes Classifier highlights is limited efficacy with a Precision of 0.54, Recall of 0.85, and F1 Score of 0.66.

7.3.2 BERT + Linear

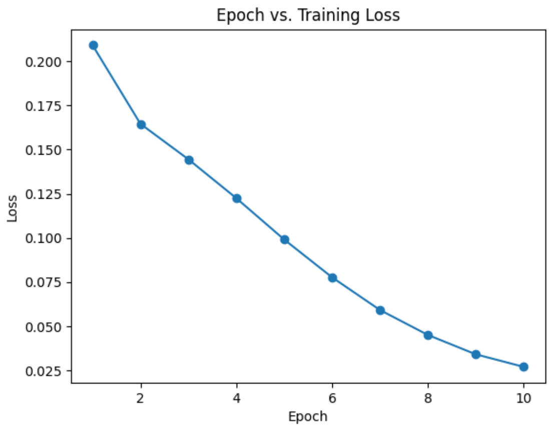


Figure 6: Plot of training loss across epoch

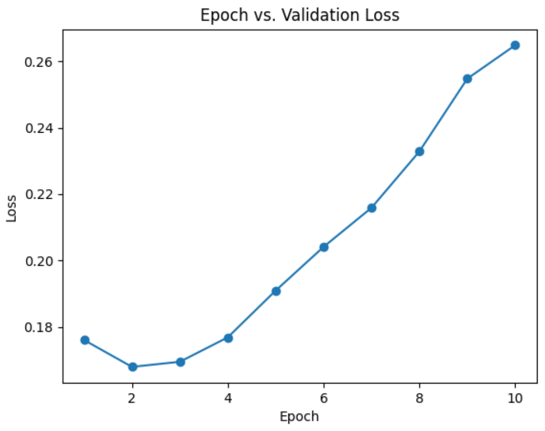


Figure 7: Plot of validation loss across epoch

The BERT-Linear classifier was trained over 10 epochs, as illustrated by the Training and Validation Loss figure. During training, the loss consistently decreased, reflecting effective learning. However, the validation loss exhibited a gradual increase after the 2nd epoch, suggesting potential overfitting. Following a thorough analysis, Epoch 2 was identified as the optimal stopping point, ensuring a balance between learning and generalization.
The conclusive performance metrics for the optimal BERT-Linear model showcase its efficacy. With a Precision of 0.85, Recall of 0.78, and F1 Score of 0.81

7.3.3 BERT + LSTM

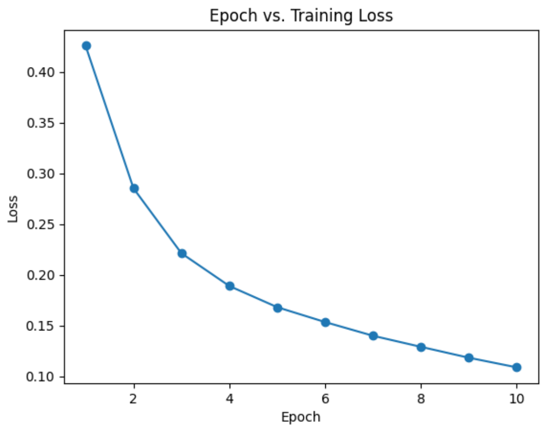


Figure 8: Plot of training loss across epoch

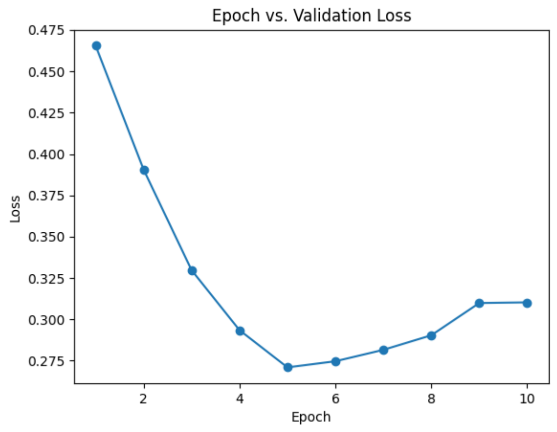


Figure 9: Plot of validation loss across epoch

The BERT-LSTM classifier was trained over 10 epochs, as illustrated by the Training and Validation Loss figure. During training, the loss consistently decreased, reflecting effective learning. However, the validation loss exhibited a gradual increase after the 2nd epoch, suggesting potential overfitting. Following a thorough analysis, Epoch 2 was identified as the optimal stopping point, ensuring a balance between learning and generalization.
The conclusive performance metrics for the optimal BERT-Linear model showcase its efficacy. With a Precision of 0.85, Recall of 0.78, and F1 Score of 0.81

7.3.4 BERT + CNN

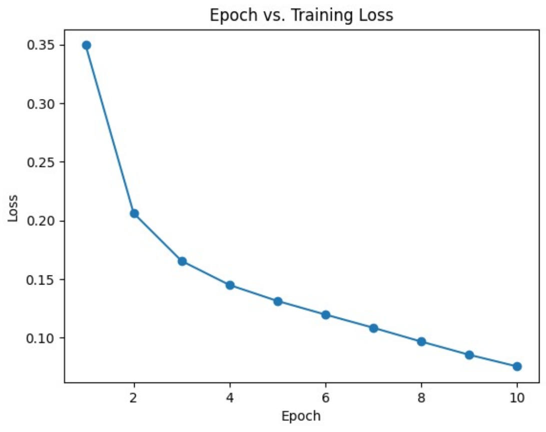


Figure 10: Plot of training loss across epoch

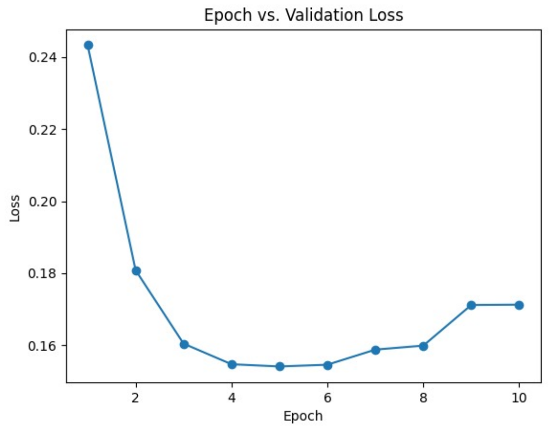


Figure 11: Plot of validation loss across epoch

The BERT-CNN classifier underwent training for 10 epochs, as depicted in the accompanying figure illustrating the Training and Validation Loss. The training loss consistently decreased throughout the epochs, indicating effective learning. However, the validation loss exhibited an upward trend, signifying potential overfitting beyond a certain point. To prevent overfitting on train data we would have to consider early stopping. After careful analysis, Epoch

6 emerged as the optimal stopping point, balancing learning and generalization.
The final Precision, Recall and F1 Score for the optimum model are 0.88, 0.88 and 0.85 respectively.

7.3.5 Model Comparison

Model	# Epochs	# Parameters	Precision	Recall	F1
BASE MODEL - MultinomialNB	-	-	54.70%	85.29%	66.65%
BERT Base - Linear Head	10	109.5M	85.15%	78.84%	81.87%
BERT Base - CNN Head	10	109.7M	88.29%	88.29%	85.18%
BERT Base - LSTM Head	10	111.6M	87.09%	82.80%	84.89%

Table 1: Result comparison across different experiments.

Table 1 shows the results obtained in our experiments. The BERT Base models are run for 10 epochs and the Train - Validation Loss curves are shown. From the table we can see that, BERT Base - Linear and BERT Base CNN have same number of parameters approximately, but CNN model outperforms the Linear head and all other models with a high Precision and Recall value of 88.29%. The optimised model has an overall F1 of 85.18%.

8 Challenges and Limitations

8.1 Resume-Job Role Multi-Label Classification

Though the selected model of BERT + CNN is able to classify resume into desirable job roles, the limited number of job roles i.e., classes pose restriction on the model. This is the limitation on classifying resumes into specific job roles. We can overcome this limitation by expanding our resume corpus to accomodate additional job roles.

8.2 Resume-Job Description Similarity Score

While RoBERTa performs well in Resume-JD similarity scoring, we can enhance the model by fine tuning the model with resume-JD dataset which helps the model adapt to the nuances of the task at hand.. We can perform additional experiment with different transformer-based models and hyperparameter settings during the fine-tuning process.

8.3 Resume-Skillset Matching

The dataset did not have resume to skills matching and hence extracting skillset from resume were not possible using a trained model. To avoid mapping file, we can train a model using the required resume-skill data yeilding better skillset extraction.

8.4 Summarization

The summarization pipeline, while effective, may not capture nuanced details in some cases. Generating highly abstractive and contextually accurate summaries remains a challenging task in natural language processing.

9 Conclusion

In conclusion, our study indicates that combining BERT with Convolutional Neural Networks (CNN) is the best choice for classifying resume into multiple job profiles. This hybrid model outperforms others by using BERT's context understanding along with CNN's ability to recognize local patterns. For Resume-Job Description (JD) similarity scoring, RoBERTa consistently performs well due to its nuanced embeddings. Additionally, our summarization pipeline proves to be a valuable tool, simplifying the analysis process and making resume content more accessible. In summary, our results provide distinct perspectives on efficient resume models, highlighting the significance of selecting models according to particular requirements.

10 Future Scope

10.1 Interpretable Models

Focusing on developing interpretable models by employing techniques such as attention visualization and saliency maps. This enhances transparency and allows users to understand how the model arrives at specific predictions.

10.2 Named Entity Recognition (NER) for Skill Extraction

Implementing advanced NER techniques for extracting specific skills from resumes, including both technical and soft skills. This could involve fine-tuning models or leveraging pre-trained models designed for NER tasks.

10.3 Optical character recognition

We can extend the initial phase of parsing pdf or docx resume to extracting text from resume which are in a image format.

10.4 Automated Feedback Loop

Establishing an automated feedback loop for continuous improvement, where user feedback on model predictions contributes to ongoing model refinement. This process will help train a better model.

10.5 Multimodal Resume Analysis

Incorporating multimodal analysis by considering not only textual information but also visual elements from resumes, such as images, graphics, and portfolios. This could provide a more holistic understanding of a candidate's qualifications.

11 Streamlit Application

The Resume Scanner Tool is developed using Streamlit, a powerful Python library for creating interactive web applications with minimal code. The tool allows users to upload resumes in PDF or DOCX format, leveraging helper functions from helperfunctions.py for parsing, classification, summarization, skillset identification, and job description comparison.

Upon uploading a resume, the tool extracts text content and displays metrics with adjustable sliders, offering users control over the job role confidence threshold for classification. The application utilizes the convert_pdf_to_txt_file function to extract text and determine predicted job roles with a confidence level. Users can visualize and interpret the predicted job roles with a dynamically adjustable confidence threshold. The tool employs Streamlit's interactive widgets, including sliders for confidence threshold adjustments and buttons for resume summarization and job description comparison. For resume summarization, the summarize_text function is utilized, providing users with concise summaries. Skillset identification is achieved using the identify_skillsets function, showcasing the skills extracted from the resume.

Furthermore, the tool allows users to input a job description for comparison with the uploaded resume. The similarity score is calculated using the compare_job_description function, offering insights into how well the resume aligns with the specified job requirements.

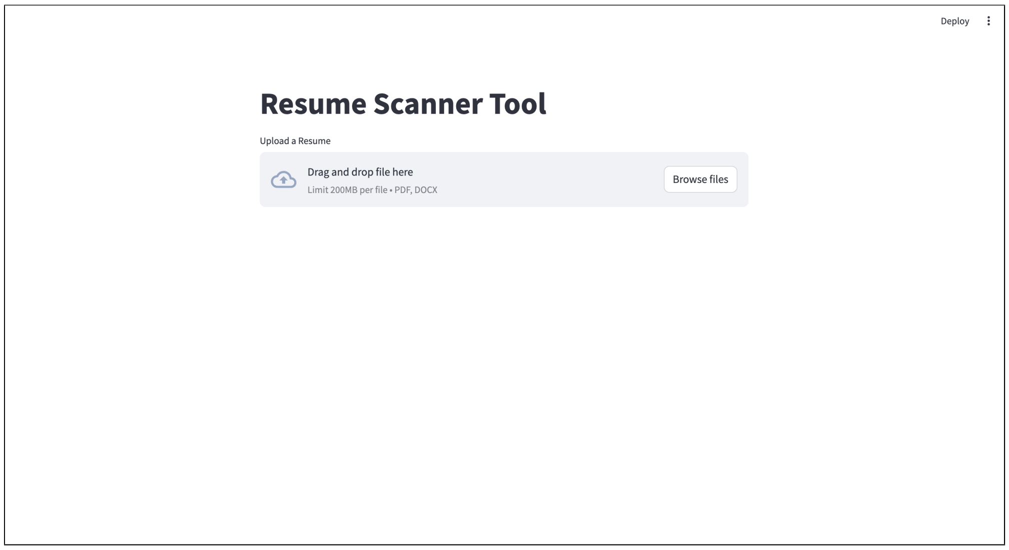
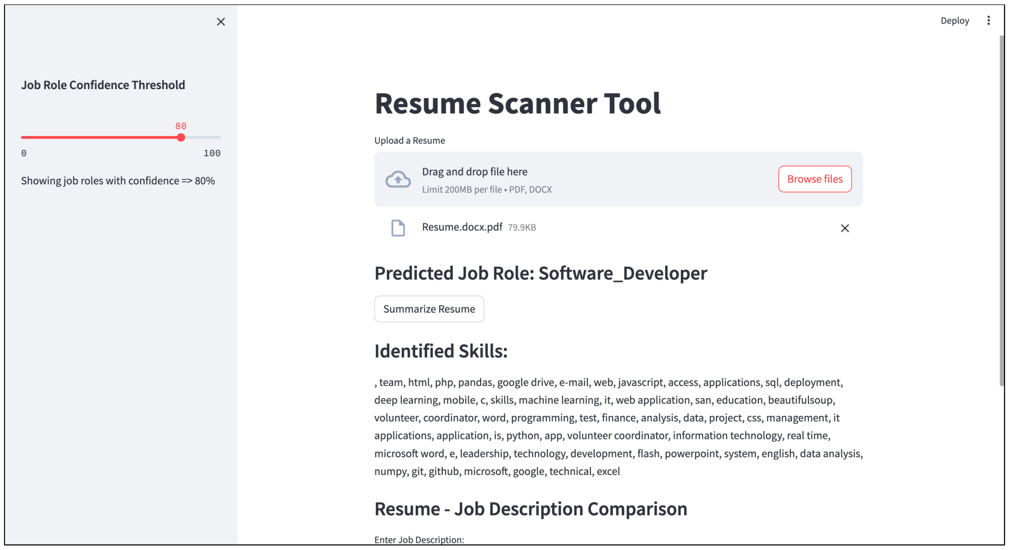


Figure 12: Landing Page of Streamlit app


Job Role Confidence Threshold

80

0 100

Showing job roles with confidence => 80%

Deploy

Resume Scanner Tool

Upload a Resume

Drag and drop file here

Limit 200MB per file • PDF, DOCX

Browse files

Resume.docx.pdf 79.9KB

Predicted Job Role: Software_Developer

Summarize Resume

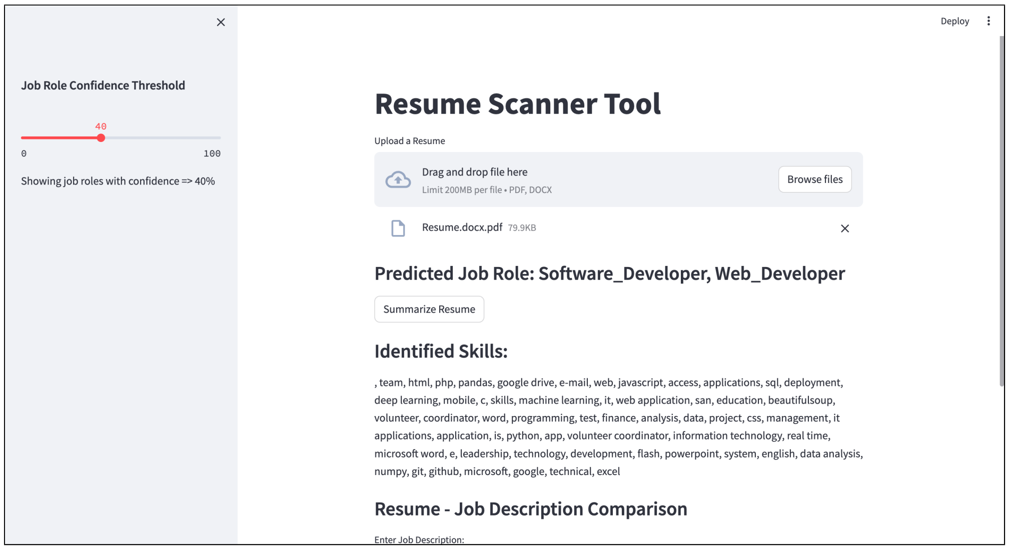
Identified Skills:

, team, html, php, pandas, google drive, e-mail, web, javascript, access, applications, sql, deployment, deep learning, mobile, c, skills, machine learning, it, web application, san, education, beautifulsoup, volunteer, coordinator, word, programming, test, finance, analysis, data, project, css, management, it applications, application, is, python, app, volunteer coordinator, information technology, real time, microsoft word, e, leadership, technology, development, flash, powerpoint, system, english, data analysis, numpy, git, github, microsoft, google, technical, excel

Resume - Job Description Comparison

Enter Job Description:

Figure 13: Predicting job roles from resume


Job Role Confidence Threshold

40

0 100

Showing job roles with confidence => 40%

Deploy

Resume Scanner Tool

Upload a Resume

Drag and drop file here

Limit 200MB per file • PDF, DOCX

Browse files

Resume.docx.pdf 79.9KB

Predicted Job Role: Software_Developer, Web_Developer

Summarize Resume

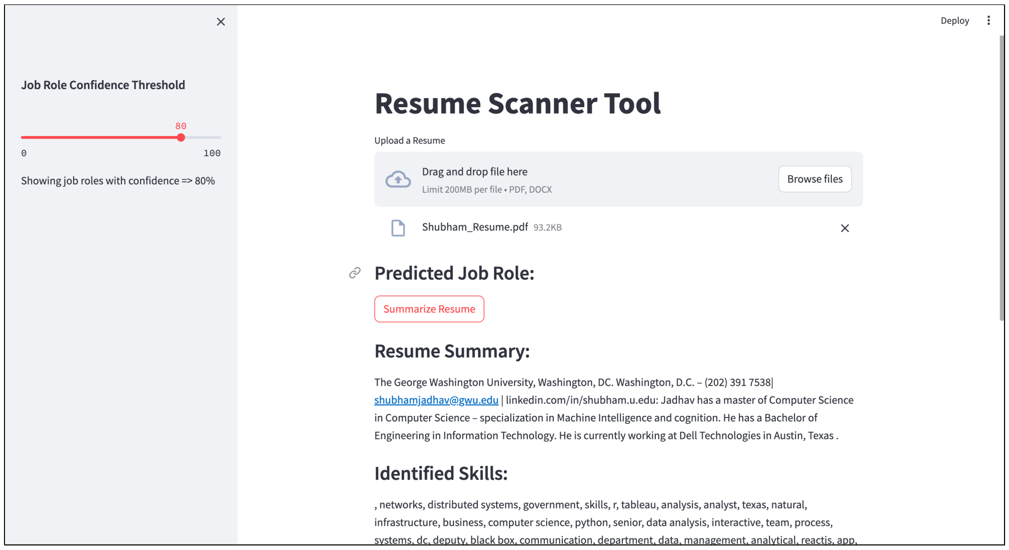
Identified Skills:

, team, html, php, pandas, google drive, e-mail, web, javascript, access, applications, sql, deployment, deep learning, mobile, c, skills, machine learning, it, web application, san, education, beautifulsoup, volunteer, coordinator, word, programming, test, finance, analysis, data, project, css, management, it applications, application, is, python, app, volunteer coordinator, information technology, real time, microsoft word, e, leadership, technology, development, flash, powerpoint, system, english, data analysis, numpy, git, github, microsoft, google, technical, excel

Resume - Job Description Comparison

Enter Job Description:

Figure 14: Predicted Job Roles with 45% threshold


Deploy
Job Role Confidence Threshold
80
0
100
Showing job roles with confidence => 80%
Upload a Resume
Drag and drop file here
Limit 200MB per file • PDF, DOCX
Browse files
Shubham_Resume.pdf 93.2KB
Summarize Resume

Resume Scanner Tool

Predicted Job Role:

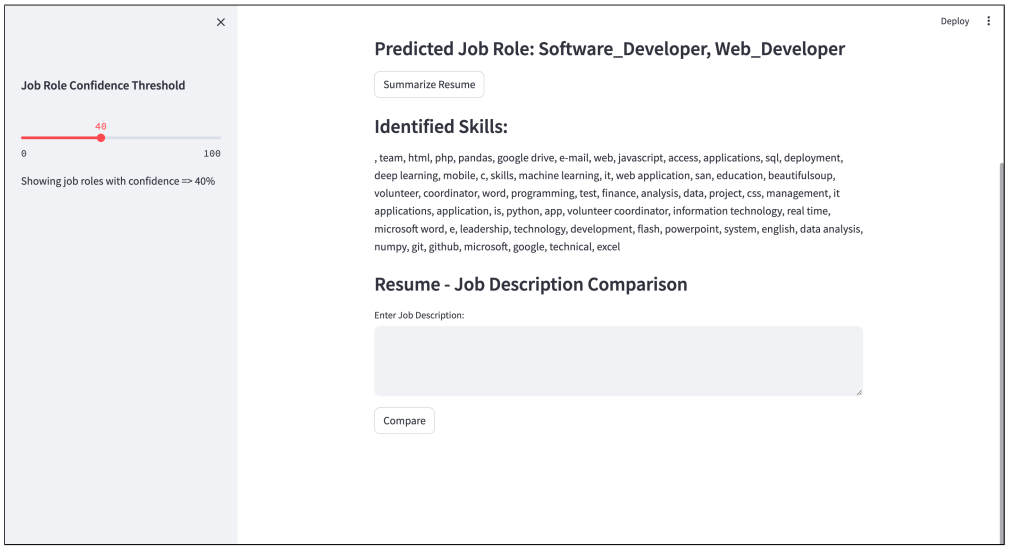
Resume Summary:

The George Washington University, Washington, DC. Washington, D.C. – (202) 391 7538| shubhamjadhav@gwu.edu | linkedin.com/in/shubham.u.edu: Jadhav has a master of Computer Science in Computer Science – specialization in Machine Intelligence and cognition. He has a Bachelor of Engineering in Information Technology. He is currently working at Dell Technologies in Austin, Texas .

Identified Skills:

, networks, distributed systems, government, skills, r, tableau, analysis, analyst, texas, natural, infrastructure, business, computer science, python, senior, data analysis, interactive, team, process, systems, dc, deputy, black box, communication, department, data, management, analytical, reactjs, app,

Figure 15: Displaying Resume summary on demand


Deploy
Job Role Confidence Threshold
40
0
100
Showing job roles with confidence => 40%
Summarize Resume
Enter Job Description:
Compare

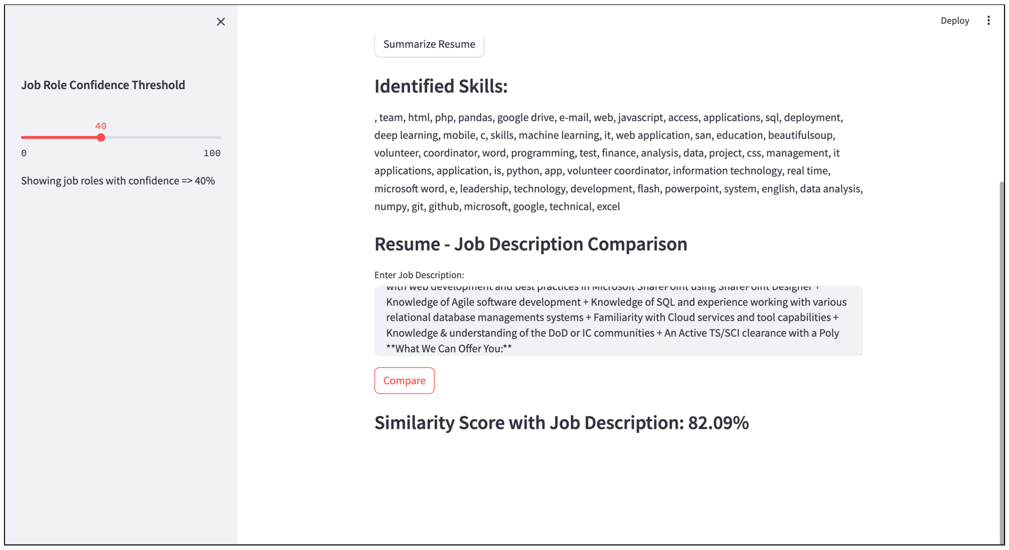
Predicted Job Role: Software_Developer, Web_Developer

Identified Skills:

, team, html, php, pandas, google drive, e-mail, web, javascript, access, applications, sql, deployment, deep learning, mobile, c, skills, machine learning, it, web application, san, education, beautifulsoup, volunteer, coordinator, word, programming, test, finance, analysis, data, project, css, management, it applications, application, is, python, app, volunteer coordinator, information technology, real time, microsoft word, e, leadership, technology, development, flash, powerpoint, system, english, data analysis, numpy, git, github, microsoft, google, technical, excel

Resume - Job Description Comparison

Figure 16: Displaying list of extracted skill set from resume



Job Role Confidence Threshold

40

0 100

Showing job roles with confidence => 40%

Identified Skills:

, team, html, php, pandas, google drive, e-mail, web, javascript, access, applications, sql, deployment, deep learning, mobile, c, skills, machine learning, it, web application, san, education, beautifulsoup, volunteer, coordinator, word, programming, test, finance, analysis, data, project, css, management, it applications, application, is, python, app, volunteer coordinator, information technology, real time, microsoft word, e, leadership, technology, development, flash, powerpoint, system, english, data analysis, numpy, git, github, microsoft, google, technical, excel

Resume - Job Description Comparison

Enter Job Description:

with web development and best practices in Microsoft SharePoint using SharePoint Designer + Knowledge of Agile software development + Knowledge of SQL and experience working with various relational database managements systems + Familiarity with Cloud services and tool capabilities + Knowledge & understanding of the DoD or IC communities + An Active TS/SCI clearance with a Poly **What We Can Offer You:**

Similarity Score with Job Description: 82.09%

Figure 17: Displaying similarity score of resume and job description

References

[1] https://huggingface.co/docs/transformers/model_doc/bart.

[2] A. Jafari. https://github.com/amir-jafari/NLP/tree/master/Lecture_09.

[3] https://huggingface.co/tasks/sentence-similarity.

[4] A. Jafari. https://github.com/amir-jafari/NLP/tree/master/Lecture_04.

[5] https://huggingface.co/docs/transformers/model_doc/bert.

[6] https://huggingface.co/docs/transformers/model_doc/roberta.

[7] E. Gomede, "Understanding multinomial naive bayes classifier." https://medium.com/@evertongomede/understanding-multinomial-naive-bayes-classifier-fdbd41b405bf, Nov 2011.