DATS 6312
Natural Language Processing

Final Term Project – Individual Project Report

Document submitted by:

Jahnavi Ramagiri

G46015075

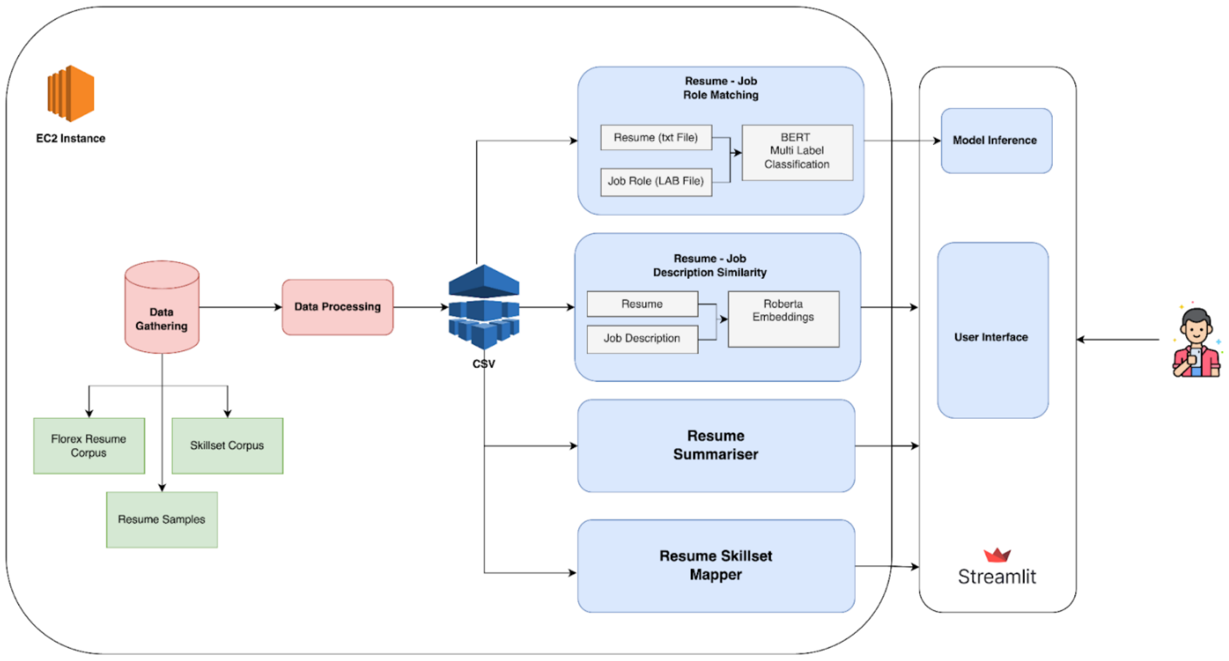
Contents

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:

- **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.
- **Multi-label Resume Classification**: Categorize resumes into predefined classes or job roles, making it easier for candidates to identify suitable job roles for their resume.
- **Resume Analysis**: Provide comprehensive analysis and insights into a candidate's resume, including key skills, qualifications, and experience.



2. Description of Individual Work

Individually, my main areas of focus were:

- **Tool Features:**
  - Actively participated in brainstorming sessions to identify and finalize features for the Resume Scanner Tool.
- **Data Cleaning (Preprocessor.py):**
  - Developed the data cleaning and pre-processing module for skillset and normalized classes encapsulated in the "Preprocessor.py" file.
- **Resume-Job Role Classification:**
  - Implemented the BERT model with CNN and LSTM classifier heads for accurate classification of resumes into job roles.
- **Skillset Matching:**
  - Curated a master skillset corpus by extracting and mapping skills from the normalized classes dataset.
- **Streamlit UI Development**:
  - Played a significant role in designing and implementing the Streamlit-based UI for the Resume Scanner Tool.
  - Contributed to the foundational code of the UI to meet project requirements.
- **Model Optimization and Hyperparameter Tuning:**
  - Explored and contributed to understanding model optimization and hyperparameter tuning.
  - Participated in determining the most suitable model for the specific use case.

2.1.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.

2.1.1. Skillset

The clean_skillset method processes individual skillsets by:

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

2.1.2. Normalised Classes

The clean_skills_norm method processes individual skillsets by:

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

- Sorting the list of skills alphabetically.
- Presenting the final cleaned skillset as a comma-separated string of individual skills.

2.2.Multi Label Classification –

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.

2.2.1. LSTM Head

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.

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.

2.2.2. CNN heads

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.

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.

2.3. Skillset Extraction

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.

2.4.Streamlit UI

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.

Individually I have worked on putting up the base code for the basic UI design of the tool.

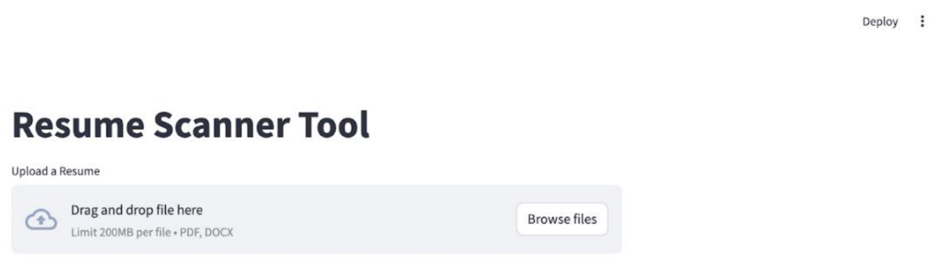
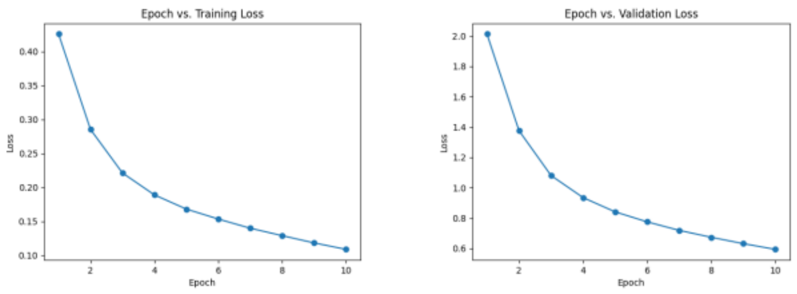


Figure shows the opening page of the resume scanner tool.

3. Results

3.1. BERT – LSTM

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.

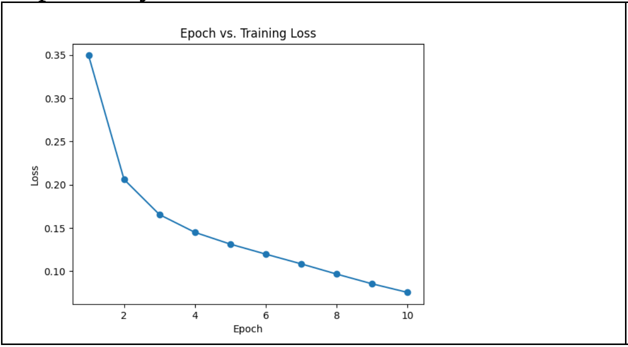


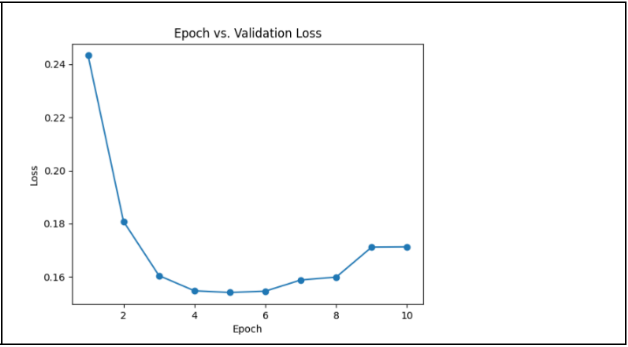
The conclusive performance metrics for the optimal BERT-Linear model showcase its efficacy. With a Precision of 0.85, Recall of 0.78, and F1 Score of 0.81

3.2.BERT – CNN

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 0.86, 0.83 and 0.84 respectively.





3.3.Final Results

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%.

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%

4. Summary and Conclusions

In summary, this project focuses on enhancing the efficiency of the job application process through the development of the Resume Analysis tool. My Key contributions include the implementation of advanced NLP techniques, such as BERT models with CNN and LSTM classifier heads, for accurate job role classification and resume analysis. Additionally, skillset extraction and a streamlined Streamlit-based UI were developed to provide a comprehensive tool for both candidates and recruiters. The results demonstrate the effectiveness of the BERT-CNN model, which outperforms other approaches with a high Precision and Recall of 88.29%. Overall, the project successfully addresses the challenges in resume analysis, providing a valuable resource for optimizing the job application experience.

In conclusion, the project achieves its objectives by combining sophisticated NLP models, efficient data preprocessing, and an intuitive user interface. The robustness of the tool is evident in the comparative results, with the BERT-CNN model showcasing superior performance. The individual contributions, ranging from model development to UI design, collectively contribute to a comprehensive solution for resume analysis and job role classification. Future work may involve further optimization, expansion of features, and potential integration with recruitment systems for broader applicability.

5. Percentage Of Code

35%

References

1. https://github.com/florex/resume_corpus/tree/master
2. https://github.com/mikeasilva/data-scientist-skills/blob/master/skills_list.txt
3. https://www.kaggle.com/datasets/wahib04/multilabel-resume-dataset
4. https://huggingface.co/transformers/v2.9.1/pretrained_models.html
5. https://arxiv.org/pdf/1910.03089.pdf