

END TERM REPORT

Shreyansh Modi 24115141 (HireSense)

Section 1: Abstract

The sole aim of my Machine Learning project is to build a candidate selection model. The model will shortlist most appropriate candidates for a particular Job Description among a huge collection of candidates. My model comes under the NLP (Natural Language Processing) domain of Machine Learning. My model utilizes various NLP techniques ranging from basic NLP techniques for text preprocessing to advanced NLP techniques for generating embeddings. I have used NLP techniques such as StopWord Removal, tokenization, lemmatization for processing the input. I will make use of mainly SpaCy to implement these. A large part of my model is based on the Named Entity Recognition (NER) which is used for extracting important fields (which earlier I was planning to do through RegEx) from Resumes and JDs which will make use of SpaCy's blank English model for implementation. Earlier I was using the Doc2Vec approach (which is an extension of the famous Word2vec developed by Google) for generating embeddings. But due to low accuracy scores I have finally used the pre-trained T5 encoder-decoder model for generating embeddings of the parsed Resumes and Job Descriptions due to its high accuracy. Earlier I was determined to use TF-IDF for generating embeddings but as suggested by my mentors and also during the learning phase I got to know about Word2Vec and dropped the idea of using it. The last part of my model includes generating similarity scores and I have used cosine similarity to check the similarity between embedded Resumes and Job Descriptions. I have used two pre-trained BERT large models fine-tuned for NER tasks: one for Parsing Resumes and the other one for parsing Job Descriptions. Parsed Resumes and JDs from those models will be fed to the T5 model to generate embeddings and finally similarity score will be calculated.

Section 2: Methodology

1. Problem Statement

To build a Candidate Selection model based on Machine Learning that would shortlist the most suitable candidate for a particular Job Description based on their Resumes. The problem is relevant since it is a very time-consuming task to manually look at all the resumes and shortlist the most appropriate one.

2. Data Collection and Preprocessing

- **Data Sources:**
 - **Used For Training Resume Parser:** A collection of around 200 annotated resumes available on [Kaggle](#) for NER tasks and a few other resumes which were manually annotated from enhancv.com to make the dataset diverse.
 - **Used For Training Job Desc Parser:** A collection of 2 lakh Job Descriptions available on Kaggle of which around 7000 were used for training.
- **Preprocessing Steps:** Tokenized all the training data followed by stop word removal and removal of punctuation marks followed by Lemmatization and finally took all the unique words to train both the models.
- **Algorithms and Techniques**
 - **NER:** Used NER to extract all the relevant information from Resumes and Job Descriptions. I had the option to use RegEX to extract these but due to lack of any pattern in Resumes this couldnt work.
 - **Fine Tuning** pre-trained Bert Large Cased Model for NER task. I have also tried the NER pipeline provided by Spacy(which gave me very low accuracy) and Bert Base Cased. BERT was chosen to capture semantic meaning due to its Transformer based architecture.
 - **Cosine Similarity** was used to calculate similarity between the embeddings of Resumes and Job Descriptions.

3. Model Development

- **Architecture Details:**
 - **Resume and JD Parser:** based on bert large cased having 24 layers

```
args = NERArgs()

args.num_train_epochs = 19

args.learning_rate = 4.464733792261065e-05

args.train_batch_size = 16

args.eval_batch_size = 8

args.gradient_accumulation_steps = 16

args.warmup_ratio = 0.08876847289936485

args.weight_decay = 0.00023027403760175153

args.evaluation_strategy = "steps"
```

```
args.eval_steps = 100

args.logging_steps = 50

args.save_strategy = "steps"

args.save_steps = 500

args.load_best_model_at_end = True

args.early_stopping_patience = 3

args.early_stopping_threshold = 0.01




















args.overwrite_output_dir = True

args.dropout=0.40358622263304433
```

The hyperparameters for both the models were tuned using Optuna.

- **Pre Trained T5 XXL:** resumes and jds annotated by above model will be fed to this to generate word embeddings which will be used to calculate cosine similarity.
- **Optimization:** Used Optuna for tuning the hyperparameters of the Resume and JD Parser Model.
- **Evaluation:** Metrics such as Precision Recall F1 Score and accuracy was used to evaluate the model.

These were the losses after each epoch(in the picture):

```
Epochs 1/19. Running Loss: 3.3885: 100%  161/161 [00:43<00:00, 3.66it/s]
/usr/local/lib/python3.11/dist-packages/simpletransformers/ner/ner_model.py:782: FutureWarning: `torch.cuda.amp.
with amp.autocast():
Epochs 2/19. Running Loss: 2.4840: 100%  161/161 [00:43<00:00, 3.50it/s]
Epochs 3/19. Running Loss: 1.9750: 100%  161/161 [00:44<00:00, 3.61it/s]
Epochs 4/19. Running Loss: 1.6597: 100%  161/161 [00:44<00:00, 3.52it/s]
Epochs 5/19. Running Loss: 1.4192: 100%  161/161 [00:44<00:00, 3.58it/s]
Epochs 6/19. Running Loss: 1.5050: 100%  161/161 [00:44<00:00, 3.54it/s]
Epochs 7/19. Running Loss: 1.1077: 100%  161/161 [00:44<00:00, 3.59it/s]
Epochs 8/19. Running Loss: 0.7024: 100%  161/161 [00:44<00:00, 3.56it/s]
Epochs 9/19. Running Loss: 0.8564: 100%  161/161 [00:44<00:00, 3.56it/s]
Epochs 10/19. Running Loss: 1.2444: 100%  161/161 [00:44<00:00, 3.56it/s]
Epochs 11/19. Running Loss: 0.5477: 100%  161/161 [00:44<00:00, 3.55it/s]
Epochs 12/19. Running Loss: 0.3523: 100%  161/161 [00:44<00:00, 3.58it/s]
Epochs 13/19. Running Loss: 1.2541: 100%  161/161 [00:44<00:00, 3.54it/s]
Epochs 14/19. Running Loss: 0.2700: 100%  161/161 [00:44<00:00, 3.59it/s]
Epochs 15/19. Running Loss: 0.9828: 100%  161/161 [00:44<00:00, 3.51it/s]
Epochs 16/19. Running Loss: 0.8153: 100%  161/161 [00:44<00:00, 3.57it/s]
Epochs 17/19. Running Loss: 0.1110: 100%  161/161 [00:44<00:00, 3.54it/s]
Epochs 18/19. Running Loss: 0.3172: 100%  161/161 [00:44<00:00, 3.55it/s]
Epochs 19/19. Running Loss: 0.8324: 100%  161/161 [00:44<00:00, 3.57it/s]
(190, 1.0692134916022615)
```

4. Implementation Workflow:

- **Data Collection and Restructuring:** Data for training both the models was collected mainly from Kaggle and Restructured in a particular format.

```
1 Sentence #,Word,NER Label,
2 Sentence: 1,Rohit,B-Name,
3 Sentence: 2,Bijlani,I-Name,
4 Sentence: 3,JAVA,B-Designation,
5 Sentence: 4,Developer,I-Designation,
6 Sentence: 5,I-Designation,,
7 Sentence: 6,Systems,I-Designation,
8 Sentence: 7,Engineer,I-Designation,
```

- **Preprocessing and Fine Tuning the Model:** Formatted Data was cleaned, lemmatized and preprocessed and was used for fine tuning.
- **Generating Embeddings:** Parsed Resumes and JDs were converted into vectors using T5.
- **Similarity Score:** Cosine Similarity was used to calculate the similarity between Resumes and JDs.

Section 3: Results

Result of Resume Parser which returned the parsed resume of a Data Scientist in JSON Format:

```
{'Skills': ['report', 'Volunteered', 'Learning', 'cover', 'transform', 'excel', 'b', 'commit', 'Storytelling', 'set', 'demonstrate', 'make', 'trend', 'generation', 'analyze', 'Bank', 'Generation', 'CERTIFICATION', 'decision', 'dashboard', 'understandable', 'datum', 'format', 'Collaboration', 'q', 'Payton', 'development', 'Report', 'participate', 'processing', 'aspire', 'objective', 'story', 'Expert', 'datum analysis', 'SQL', 'data', 'Hands', 'operational', 'Skilled', 'Food', 'drive', 'enhance', 'stakeholder', 'LANGUAGES', 'English', 'visualization', 'team', 'develop', 'manipulation', 'management', 'Training', 'making', 'workshop', 'Spanish', 'training', 'depth', 'optimize', 'intelligence', 'Business', 'Webster', 'Eager', 'linkedin', 'creation', 'query', 'feedback', 'allocation', 'insight', 'analysis', 'automate', 'Tableau', 'pattern', 'streamline', 'Python', 'provide', 'project', 'strategy', 'system', 'evidence', 'collaborate', 'reporting', 'understanding', 'focus', 'Data', 'resource', 'Thinking', 'contribute', 'volunteer', 'Coursera', 'analytic', 'collaboration', 'communication', 'conduct', 'collection', 'Visualization', 'com', 'dataset', 'lead', 'identify', 'internship', 'business', 'support', 'result', 'process', 'apply', 'study', 'prove', 'help', 'z', 'Continuous'], 'Location': ['Cleveland', 'Cleveland OH'], 'Quality': ['positive', 'accuracy', 'eager', 'key', 'skilled', 'achieve', 'dynamic', 'drive', 'Advanced', 'complex', 'focus', 'self', 'contribute', 'large', 'Passionate', 'compelling', 'efficiency', 'increase', 'reduce', 'actionable', 'continually', 'improve', 'quality', 'improvement', 'present', 'enhance', 'Comprehensive', 'power', 'effective'], 'Companies worked at': ['IBM', 'Udemy', 'Case', 'Accenture', 'Western', 'Native'], 'Years of Experience': ['40', '2022', '2019', '15', '20', '30', '25', '10', '2023'], 'Designation': ['Bachelor Science']}
```

Result of JD Parser which returned the parsed Job Desc of a Data Scientist in JSON Format:

```
'Designation': ['Data Analyst Data Scientist', 'Data Scientists'], 'Responsibility': ['expertise', 'analysis', 'machine', 'extract', 'insight', 'prediction', 'build', 'conduct', 'analysis', 'communicate', 'drive'], 'Skill': ['statistical']}
```

Section 4: Final Deliverable

The Final Deliverable of my Project is a comparative analysis shown in the [final notebook](#) uploaded on GitHub in which i have showed the similarity scores of multiple diverse resumes including a resume of an Electrical Engineer with a Job Description of an Electrical Engineer. The Score was highest for the Electrical Engineer than any other candidate which demonstrates the purposes of my project.

```
In [85]: eval(elecengresume,elecengjd)

Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Skills Similarity: 0.7764 (Weighted Contribution: 0.2329)
Degree Similarity: 0.0000 (Weighted Contribution: 0.0000)
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Designation Similarity: 0.8746 (Weighted Contribution: 0.1749)
Experience Similarity: 0.0000 (Weighted Contribution: 0.0000)
Companies Worked At Similarity: 0.0000 (Weighted Contribution: 0.0000)
Overall Similarity: 0.4078
```

```
In [86]: eval(animatorresume,elecengjd)

Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Skills Similarity: 0.6904 (Weighted Contribution: 0.2071)
Degree Similarity: 0.0000 (Weighted Contribution: 0.0000)
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Designation Similarity: 0.7811 (Weighted Contribution: 0.1562)
Experience Similarity: 0.0000 (Weighted Contribution: 0.0000)
Companies Worked At Similarity: 0.0000 (Weighted Contribution: 0.0000)
Overall Similarity: 0.3633
```

```
In [87]: eval(bartenderresume,elecengjd)

Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Skills Similarity: 0.6801 (Weighted Contribution: 0.2040)
Degree Similarity: 0.0000 (Weighted Contribution: 0.0000)
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Batches: 0%|          | 0/1 [00:00<?, ?it/s]
Designation Similarity: 0.7328 (Weighted Contribution: 0.1466)
Experience Similarity: 0.0000 (Weighted Contribution: 0.0000)
Companies Worked At Similarity: 0.0000 (Weighted Contribution: 0.0000)
Overall Similarity: 0.3506
```

Section 5: Challenges Faced

Many Challenges were faced during the making of whole project these are described below:

- **Resume Parsing:** This was the major challenge of the project. It required a lot of research for me to know which technique would be most suitable for entity extraction. I tried techniques like RegEx but it gave very poor results. Finally I used NER for this task.

Once I started working on NER ,I trained Spacy's blanked english pipeline for custom NER labels but that was very less accurate.Then finally I used BERT Large for NER and it gave me satisfactory results.

- **Data Collection and processing:** I couldn't find annotated Resumes for NER in the format which I wanted therefore It was a very tedious task for me to convert a huge amount of resumes as well as Job Descriptions in the format which I needed.Also the Resume dataset which I used was dominated by software resumes therefore I had to manually diversify the dataset.
 - **Computational Difficulties:** Limited usage of GPUs and TPUs in Kaggle as well as Google Colab(these are the two main platforms used during the project).
 - **Extremely Less Accuracy of Doc2Vec Model:** My trained Doc2Vec didn't gave reliable results,therefore I tried many other alternatives such as changing format of training dataset,Hyperparameter tuning,Fine tuning a pre trained model but due to less reliable results I had to use a pre trained model.
-

Section 6: Secondary Goals and Future Scope

Some of the goals which I planned to do but couldn't achieve are:

- Developing a ranking and a better way to calculate similarity scores to rank the resumes according to suitability.
- Fine Tuning a pre-trained model especially on Job Desc and Resume Dataset to improve the quality of embeddings,but couldn't be done due to time and resource constraints.
- Deploying my model on Streamlit or developing any other functional application or webpage to improve usage.

Suggestions for future improvements or extensions of the project.

- Integrating a Bias Detection Module.
 - Enhancing the quality of the Training Dataset.
 - Improving the accuracy of my Resume and JD parsing model, as well as fine tuning embeddings model.
-

Section 7: References

- Completed Andrew Ng's ML Specialization course on Coursera to get into Machine Learning.
 - [YouTube Playlist](#). I have used this YouTube playlist to understand the basic concepts of NLP like: RegEx, Stemming, Tokenization, Word Embeddings, etc.
 - Read this [article](#) on PDF parsing which mainly talks about PyPDF2 but since I wanted to parse not only PDFs but also doc and other files so I decided to use PyMuPDF.
 - I used this [article](#) for learning the implementation of PyMuPDF.
 - I mainly followed the YouTube videos of 3blue1brown for understanding Deep Learning and advanced NLP concepts of RNN, LSTMs, Attention, Transformers, BERT.
 - 3B1B Deep Learning [playlist](#) for Neural Networks.
 - I watched CodeBasics yt videos on Word2Vec to understand the underlying concept and also its implementation. [Vid1](#) , [Vid2](#)
 - Watched this [video](#) for the implementation of BERT.
 - Read an [article](#) provided by my mentor which talks about Doc2Vec and uses it in a similar way as i am going to use it for resume matching.
 - Watched this [yt video](#) to understand Doc2Vec(PV-DM and PV-DBOW) and also for the implementation.
 - Used [SpaCy's documentation](#) for learning how to add custom NER labels to a blank Spacy pipeline.
 - T5 Documentation. [Link](#)
 - Doc2Vec Documentation. [link](#)
-

Section 8: Appendix (Optional):

- RegEx was used for Parsing the Resumes but the accuracy was very low due to absence of a commonly accepted format for resumes and any pattern for extracting experiences and other fields.