RESUME PARSING AND JOB MATCHING

Jaini Gala - gala.j@northeastern.edu
Raghav Rastogi - rastogi.ra@northeastern.edu
Kartik Ullal - ullal.k@northeastern.edu
Srijha Kalyan - kalyan.sr@northeastern.edu

Abstract

Natural Language Processing is used to read and understand the information from the resume, find the keywords and relevant information and score the resume based on keyword matching. First the user needs to upload the resume in the pdf format and paste the job description into the text box. The parser extracts the relevant information from the resume and the job description and rate the resume using cosine similarity.

1. Introduction

It has been a long challenge for software developers to create resume parsing tools that are accurate, efficient and can detect all the information that recruiters need. Corporate companies and recruitment agencies process numerous resumes daily. This task is very difficult for humans. An intelligent automated system is required that can take out all the vital information from the unstructured resumes and transform all of them to a common structured format that can be scored for a specific job position.

1.1 Problem Statement

The main objective of the NLP-based Resume Parser in Python project is to create a web application that extracts the required information about candidates and performs similarity detection with job description and score the resumes.

Following are the key inputs and expected outputs.

Input : Resume PDF and Job description text

Output : Labelled data (Name, organisation, skills, etc) and Similarity Score between the job description and Resume

1.2 Motivation

The existing recruitment process is more time consuming and tedious, demanding applicants to fill out all of their skills and information manually. Besides that, the HR team requires more manpower to analyse the cv's of the candidates. As a result, we were inspired to create a seamless and automated solution. This will drastically save time to read and find the relevant information in a resume and also quickly find the relevant resumes from a large pool of applicants.

1.3 Future Scope

Our system is currently trained with a STEM-related dataset and can precisely score resumes for STEM disciplines. To make the system more general, we can train it with records from numerous domains that can score resumes from diverse sectors. Additionally, We could also retrieve the entities from the resume employing transformers or equivalent models. We can then scale the model where it can process large information and at the same time get refined as more and more resumes are uploaded get through the model.

2. Method

2.1 Dataset

The dataset has been taken from kaggle[1], it has 220 resumes of which all the resumes have been manually labelled using the dataturks annotation tool and have been converted into a json format suitable for spacy NER model.

The labels are divided into the following 10 categories:

- Name
- College Name
- Degree
- Graduation Year
- Years of Experience
- Companies worked at
- Designation
- Skills
- Location
- Email Address

2.2 Architecture

2.2.1 NER Model

We perform NER on the resume, to extract all the skills and important information such as organisations worked at, education institutes, Name, address, etc. This is done using the below processing pipeline. We use Spacy for training in which the NER system contains a word embedding strategy using sub word features and a deep convolutional neural network with residual connections.

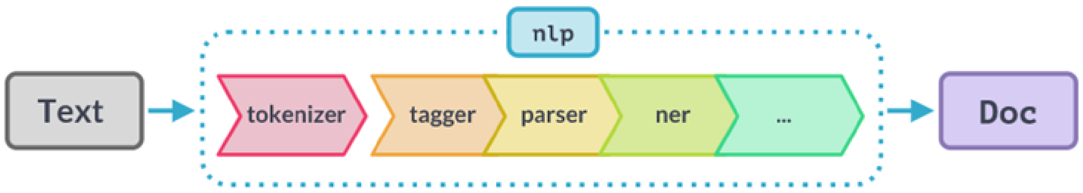


Figure 2.1: NER Process

Following are the components and their purpose in the pipeline:

- Tagger - Assigns part-of-speech tags.
- Parser - Assigns dependency labels
- NER - Detect and label named entities

As the doc moves through the pipeline, Spacy uses the pretrained model and the current training data to recognize the entities

2.2.2 Similarity Detection

For similarity detection, the first thing we had to do was convert the skills that were extracted from the resume and the job description into sentence embeddings. We use sentence embeddings instead of word embeddings because the skills can be of multiple words. For example, a skill, *Natural Language Processing,* is a multiple word skill, and a word embedding will get us separate embeddings for *Natural, Language, and Processing.* Which will consequently not give us accurate similarity results for job description and resume skills. Instead, a sentence embedding model will give us one single embedding for the skills of natural *language processing.*

For sentence embeddings, we used two different models. The first model was the Doc2Vec model (Mikolov and Le [5]). and the second model we used was a sentence transformer called the Sentence-Bert (Reimers and Gurevych #) (Reimers and Gurevych [6]).

Doc2Vec:

Doc2Vec is very similar to Word2Vec, but instead of providing word embeddings, it provides one single embedding for a document of variable size. In our Paragraph Vector framework, every paragraph is mapped to a unique vector, represented by a column in matrix D and every word is also mapped to a unique vector, represented by a column in matrix W. The paragraph vector and word vectors are averaged or concatenated to predict the next word in a context.

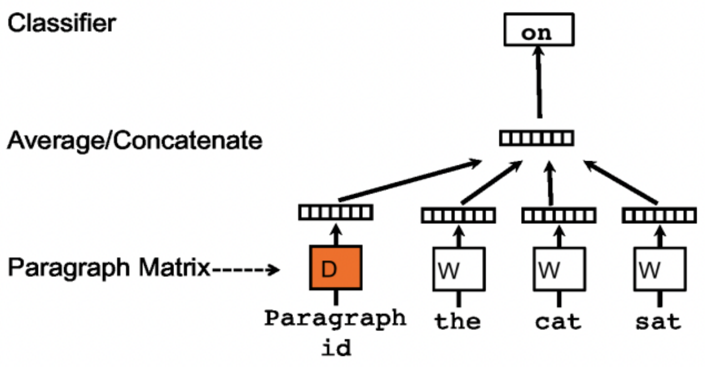


Figure 3.1 Doc2Vec model

The contexts are fixed-length and sampled from a sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph but not across paragraphs. The word vector matrix W, however, is shared across paragraphs. I.e., the vector for "powerful" is the same for all paragraphs. The paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation. At prediction time, one needs to perform an inference step to compute the paragraph vector for a new paragraph. This is also obtained by gradient descent.

Sentence-Bert (SBERT):

SBERT adds a pooling operation to the output of BERT / RoBERTa to derive a fixed sized sentence embedding. There are two pooling strategies: Using the output of the CLS-token, computing the mean of all output vectors (MEANstrategy), and computing a max-over-time of the output vectors (MAX-strategy). BERT is fine-tuned on the siamese and triplet networks (Schroff et al., 2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

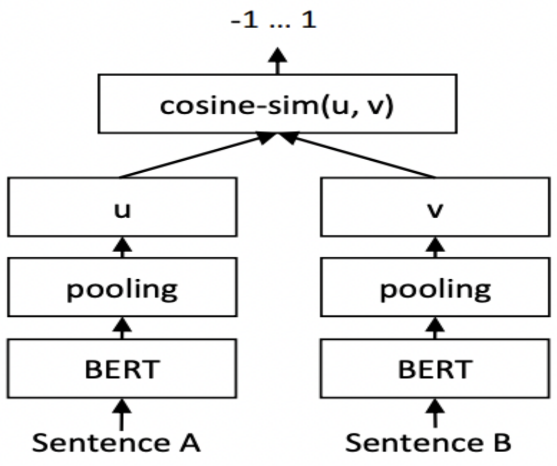


Figure 3.2: SBERT Architecture

We train SBERT on the combination of the SNLI and the Multi-Genre NLI dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels contradiction, eintailment, and neutral. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text.

Once we have the sentence embeddings from our two models, we can simplity calculate the similarity score using cosine similarity.

Cosine Measure:

The cosine—like most measures for vector similarity used in NLP—is based on normalized dot product. This normalized dot product turns out to be the same as the cosine of the angle between the two vectors.

$$\mathbf{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Figure 3.3: Cosine Measure

The cosine value ranges from 1 for vectors pointing in the same direction, through 0 for orthogonal vectors, to -1 for vectors pointing in opposite directions. But since raw frequency values are non-negative, the cosine for these vectors ranges from 0–1.

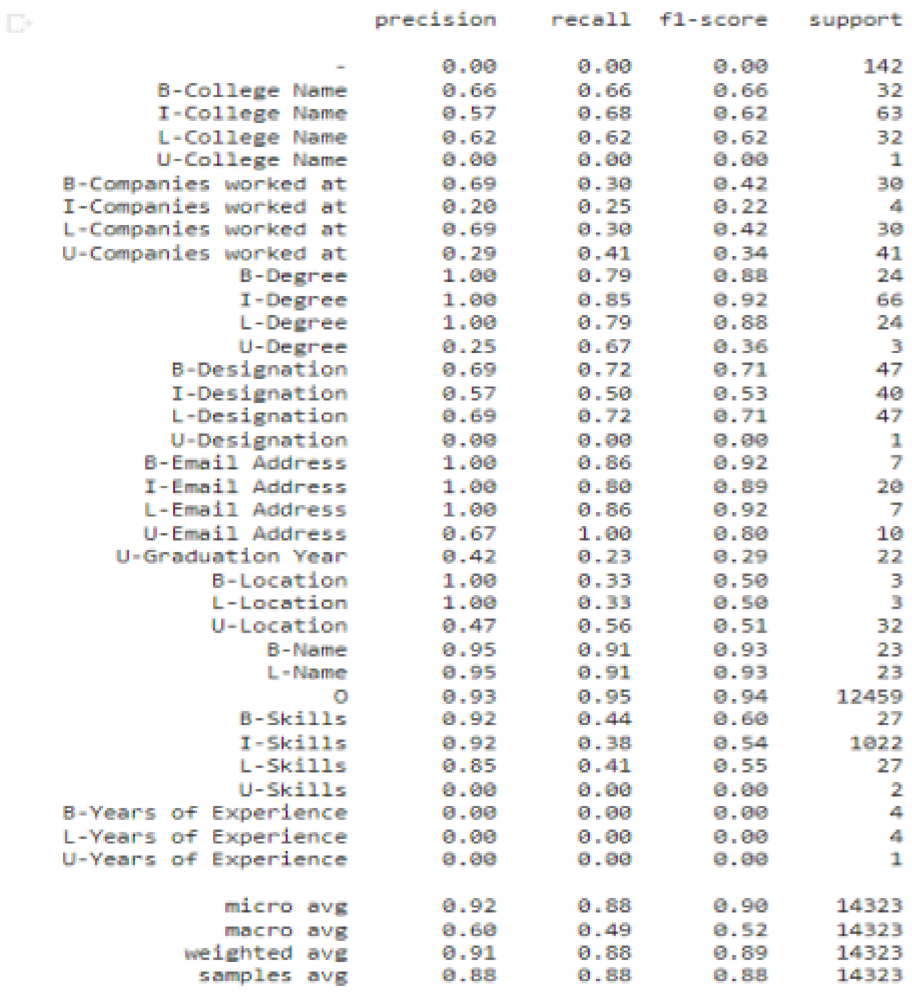
3. Results

3.1 NER Model

We evaluate the NER model from the attached classification report of the words. Here the label “B”, “I”, “L”, “O” and “U” stands for the following :

B - Words that are in the 'Beginning' of Named Entity
I - Words that are in the 'Inside of Named Entity
L - Words that are in the 'Last' of Named Entity
O - Words that are in the 'Outside' of Named Entity
U - ‘Unit’ length of Named Entity

We used sklearn classification matrix to plot the precision, recall, F1-score and support.



```
                           precision    recall  f1-score   support

                      -       0.00      0.00      0.00       142
         B-College Name       0.66      0.66      0.66        32
         I-College Name       0.57      0.68      0.62        63
         L-College Name       0.62      0.62      0.62        32
         U-College Name       0.00      0.00      0.00         1
    B-Companies worked at       0.69      0.30      0.42        30
    I-Companies worked at       0.20      0.25      0.22         4
    L-Companies worked at       0.69      0.30      0.42        30
    U-Companies worked at       0.29      0.41      0.34        41
               B-Degree       1.00      0.79      0.88        24
               I-Degree       1.00      0.85      0.92        66
               L-Degree       1.00      0.79      0.88        24
               U-Degree       0.25      0.67      0.36         3
          B-Designation       0.69      0.72      0.71        47
          I-Designation       0.57      0.50      0.53        40
          L-Designation       0.69      0.72      0.71        47
          U-Designation       0.00      0.00      0.00         1
        B-Email Address       1.00      0.86      0.92         7
        I-Email Address       1.00      0.80      0.89        20
        L-Email Address       1.00      0.86      0.92         7
        U-Email Address       0.67      1.00      0.80        10
      U-Graduation Year       0.42      0.23      0.29        22
             B-Location       1.00      0.33      0.50         3
             L-Location       1.00      0.33      0.50         3
             U-Location       0.47      0.56      0.51        32
                 B-Name       0.95      0.91      0.93        23
                 L-Name       0.95      0.91      0.93        23
                      O       0.93      0.95      0.94     12459
               B-Skills       0.92      0.44      0.60        27
               I-Skills       0.92      0.38      0.54      1022
               L-Skills       0.85      0.41      0.55        27
               U-Skills       0.00      0.00      0.00         2
    B-Years of Experience       0.00      0.00      0.00         4
    L-Years of Experience       0.00      0.00      0.00         4
    U-Years of Experience       0.00      0.00      0.00         1

              micro avg       0.92      0.88      0.90     14323
              macro avg       0.60      0.49      0.52     14323
           weighted avg       0.91      0.88      0.89     14323
            samples avg       0.88      0.88      0.88     14323
```

Figure 4.1: Shows the results of the classification report for NER

We can plot the NER loss over each epoch. Figure 4.3 shows how the loss has droppedfrom 25000 from epoch 0 to almost 3000 at epoch 5, while training the model.

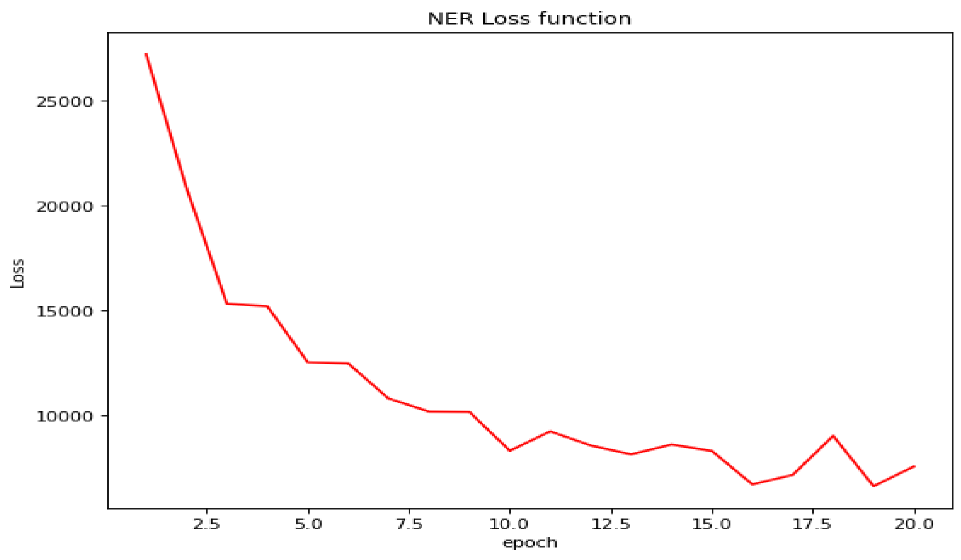
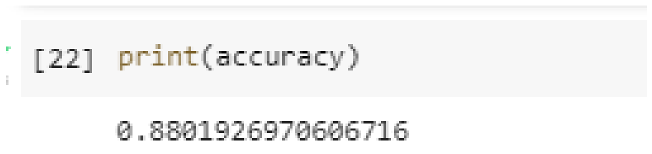


Figure 4.2: NER loss over each epoch

We also used sklearn's accuracy_score to calculate the accuray of our predicted entities with the true entities. Sklearn's accuracy_score computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true.



```
[22] print(accuracy)

     0.8801926970606716
```

Figure 4.3: Shows accuracy of the NER model

As you can see from figure 4.2, the NER model gives an accuracy of 88%, which is a fairly good accuracy score. We can improve our accuracy by using better models like the encoder-decoder transformers.

3.2 Similarity Detection

For similarity detection we used cosine similarity to give us the similarity score on two different types of embeddings. We used a data science resume, and two different job descriptions, one for data science and the other for software engineering.

Here are the results from the similarity detection.

	Data Science	Software Engineering
Doc2Vec	78%	35%
Sentence-BERT	92%	78%

As you can see, the Doc2Vec model gives varied results for the data science resume and the job descriptions. This is because the Doc2Vec model is trained on the resume skills from the dataset. Sentence-Bert gives a high similarity for both the job descriptions because it is trained using the NLI and MNLI dataset which contains almost 570,000 and 430,000 sentence pairs respectively.

4. Discussion

The NER model recognizes the important entities from a test resume and outputs the information in a usable format without going through the resume manually. The test accuracy of around 88% shows there is scope for improvement given more computational resources and more data. The classification report displays the weaknesses and strengths of the model and can be used in training underrepresented values more to improve specific entities.

Sentence Embeddings in the similarity detection can be improved by fine tuning the Sbert model, and also by separating each skill from the list of skills for the embeddings, instead of concatenating all skills into a string for one embedding. This might improve the variance in each embedding, and might give an a accurate similarity score using the cosine similarity measure.

5. Conclusion

The training resumes are manually annotated using the dataturks annotation tool. A user's résumé and job description are analysed using the Name Entity Recognition model to retrieve various items. The cosine similarity method algorithm compares an individual's skills and expertise to the job requisite skills when analysing resumes. Currently, the system ranks resumes precisely for STEM disciplines, but in the long run, we will be able to train for different domains.

6. Acknowledgement

Implementing the project titled ‘Resume Parser and Job Matching’ was a great learning experience. Right from envisioning the project till building the final project it took a lot of experimentation and iteration, we not only learnt the concepts of Natural Language Processing, but also learnt a lot about Time Management, Work Distribution, TeamWork and Project Distribution.

We are particularly grateful to Prof. Uzair Ahmad for his invaluable guidance, constructive insights and expertise that played a vital role in the advancement of this project.

7. References

1. https://www.kaggle.com/datasets/dataturks/resume-entities-for-ner
2. https://deepnote.com/@abid/spaCy-Resume-Analysis-gboeS3-oRf6segt789p4Jg
3. https://alwaysbelearning.nl/matching-resumes-with-job-offers-using-spacy-a-natural-language-processing-nlp-library-in-python/
4. https://www.youtube.com/watch?v=HJy11kOlgvk&t=2s
5. Mikolov, Thomas, and Quoc V. Le. “Distributed Representations of Sentences and Documents.” www.arxiv.org, PMLR, 22--24th June 2014, http://proceedings.mlr.press/v32/le14.pdf . Accessed 24 April 2022.
6. Reimers, Nils, and Iryna Gurevych. *Sentence-{BERT}: Sentence Embeddings using {S}iamese {BERT}-Networks*. Hong Kong, Association for Computational Linguistics, 2019. *https://aclanthology.org/*, https://aclanthology.org/D19-1410. Accessed 24 April 2022.