Artificial Intelligence CPT_S 540

Washington State University

Project Write-up

Resume/CV Parsing with NLP

Team Members:

Harshitha Girish – 011809582
Mahesh Kumar Srinivas - 011845961
Sharath Kumar Karnati - 011852253
Sreehari Guruprasad – 011809587

Repo Link : https://github.com/SreehariG73/AIProject

Table of Contents

Introduction

For every job opening in a single company, hundreds of resumes land on HR desks. Such a huge amount of information is hard to sort out and is a huge burden for the recruiter, which in turn makes the whole process of selecting a candidate slower. Each resume comes in different sizes and formats making it visibly amusing but just adds to the chaos. This can be solved using an amazing tool – resume parser/ analyzer.

Resume parser converts and transforms vast amounts of information and resumes into a much more concise, clear, and organized format. It is a tool that extracts the required information or the key details from a resume. Details like name, skills, education, experience. This conversion of free form data to more organized and structed data makes HR's job easier. This not only helps HR get the key details, but it also gives an ample amount time for the recruiter to do his job and evaluate the candidate against specific criteria.

This tool benefits beyond recruiting process. It helps the job seekers as well. Job seekers use this tool to get insights from their resume analysis. Resume parser helps in identifying mistakes, suggests the required skills and edits to be made. This tool acts like an overall quality assessment and helps in enhancing the job seekers resume. Our project's main aim is to build a tool – resume parser using powerful tech like NLP and deploy it to a machine learning model. The tool will automatically pull-out crucial data from all kinds of resumes and organize them accurately.

Problem Statement:

The conventional way of selecting and going through resumes to select candidates is both time consuming and can also be prone to human errors. A HR or recruiter should invest a considerable amount of time and energy in extracting crucial information which in turn delays the recruiting process. Therefore, our objective is – creating a tool that will automatically extract crucial information from resumes like contact information, education, experience, skills etc. The purpose of this project is to support both recruiters as well as candidates throughout the job process, making sure their time and energy is allocated towards more productive tasks.

Data Collection and Pre-processing:

For this project, we are using a dataset that consists of 2400+ resumes.

1. **Name:** Resume Dataset
2. **Source:** Kaggle

We worked with a dataset comprising over 2400 resumes. The plan was to treat each resume as a string and extract the raw data by removing all stop words. This process enabled us to gather meaningful significant information from resumes without the noise of common words that didn't carry much.

1. Text Extraction:
   Extracting raw text from the PDF files using PyPDF2 library, enabling further processing and analysis.
2. Conversion to Lowercase:
   Converting text to lowercase to standardize the word representation and avoid case sensitivity.
3. Removal of Special Characters and Punctuation:
   Eliminating special characters and punctuation to streamline text data and reduce noise.
4. Tokenization:
   Breaking down text into individual words or tokens to facilitate analysis at the word level.
5. Stop word Removal:
   Removing common words like "the" and "and" to enhance the relevance of text features.
6. Lemmatization:
   Reducing words to their base form to reduce dimensionality and improve model performance.
7. Vectorization:
   Converting text data into numerical representations (feature matrices) for machine learning model compatibility.

NLP Implementation:

Natural Language Processing:

Natural Language Processing is a subtopic in Artificial Intelligence which helps in interaction and communication between computers and human beings. NLP helps computers understand human language; It interprets in such a way that it can be understood.

Why NLP?

- NLP is useful while extracting information from unstructured data. In our case, it helps in extracting useful information from all the resumes.
- NLP analyses text data and gives patterns and insights which helps in decision making. In our project it helps when Users want to gain useful insights of their resumes when compared with job description.
- It also helps in automating tasks such as - document summarization, text classification and many more tasks. So, it is used for automation increasing efficiency and productivity.
- NLP helps with personalization. (i.e.) NLP algorithms can analyze user preferences and deliver personalized experiences.
- NLP helps in language translation. It can automatically translate different languages making communication easy.

Spacy:

It is an open-source library for Natural Language Processing tasks and is in python language. It is mainly designed for production ready NLP applications.

Features of Spacy:

- Tokenization:
  Spacy helps in tokenization making it easy to split text into different tokens efficiently. In our project, we use this feature to split the text in resume and extract useful information from different formats easily.
- Part of Speech Tagging:
  Once tokenization is done, it also helps in identifying parts of speech. It can tag each token or word into noun pronouns adjective etc. For example, in our project, every resume has different University and Names; this can be tagged as pronouns easily using spacy.
- Dependency parsing:
  Spacy also includes dependency parsing. This feature helps in identifying the relationship between different words in a document.
- Named Entity Reignition:
  It includes pre trained models for NER, which makes it easy to identify and classify named entities.
- Integration:
  Spacy library seamlessly integrates with other python frameworks and libraries. This makes it easy to incorporate NLP functionality in our application.

Cosine Similarity for Finding Similarity between Resume and Job description:

Generally, cosine similarity is used to find similarity between two multidimensional vectors. In case of our project, we are using NLP, so cosine similarity is employed to score the similarity between a job description and respective job description.

Cosine Similarity in job Matching:

It is used to compare job descriptions with resumes to select the most relevant candidates for a given job. Working of cosine Similarity:

1. Vector representation
   Each job description/ resume is represented as a vector. Where each vector is a in multidimension corresponding to unique terms in vocabulary.

2. Term frequency Inverse Document Frequency representation
   This mainly accounts for the frequency of the words that occurred in a document. This helps in recognizing importance of terms and their relationships to the entire corpus.

3. Computing Cosine similarity:
   This is calculated by analyzing the cosine angle between the two vectors. Here dot product of the two vectors is divided by the product of the magnitudes.
   cosine_similarity $(a, b) = a \cdot b \|a\| \|b\|$cosine_similarity(a, b)=‖a‖‖b‖a·b

4. Interpretation:
   Finally, the cosine scores are interpreted. The score range is between -1 to 1 where,
   - 1 – indicates perfect similarity,
   - 0 – indicates no similarity,
   - -1 – indicates complete dissimilarity.

Modeling and Training:

- Our research began its data journey with the acquisition of a large dataset comprising of over 1,000 resume PDFs received from Kaggle. This first data set served as the basis for our investigation. To prepare this data for relevant insights, we used several preprocessing approaches described in Section 3 of our methodology.

- During preprocessing, our primary goal was to enhance and standardize the data, effectively reducing noise and fixing formatting issues. This careful process was critical in converting the raw data into a structured and appropriate format for in-depth examination.

- Following preprocessing, we diligently completed a rigorous data cleaning procedure. This process was critical for removing erroneous and unnecessary data points, maintaining the dataset's integrity and trustworthiness. By rigorously cleaning the data, we improved its quality, laying the groundwork for subsequent analysis and modeling.

- With a revised and clean dataset in hand, we began feature extraction. This entailed extracting the most important qualities from each resume and dividing them into distinct groups based on job titles such as Data Science, Software Engineer, HR, Associate, Sales, and others. This segmentation allowed for a better understanding of the dataset's structure and targeted analysis based on specific employment types.

- After formatting the data to meet our analysis needs, we used the k-Nearest Neighbors (KNN) algorithm to train the model. This method, known for its simplicity and efficacy in classification problems, was an excellent fit for our categorization goal.

- To evaluate our model's performance, we used a rigorous evaluation procedure that included parameters such as accuracy, precision, recall, and F1-score. These metrics offered a complete summary of our model's ability to effectively categorize resumes into their appropriate job categories.

- Our results showed a high level of accuracy, precision, recall, and F1-score, demonstrating the durability and efficacy of our approach for effectively categorizing resumes. This successful outcome confirmed the efficacy of our approach and the soundness of our analytical procedure.

Then we are going to apply Cosine similarity:

- We will utilize the preprocessed data from our previous step, which has been normalized and transformed into a numerical representation. We will then apply the cosine similarity algorithm to compare each resume with the job description. This algorithm calculates the dot product of the two vectors (resume and job description) and divides it by the product of their magnitudes, resulting in a score between 0 and 1.
- This score represents the degree of similarity between the resume and the job description, with higher scores indicating a closer match. The cosine similarity score considers the importance of each word in the job description and resume, allowing us to capture the text's semantic meaning.
- By obtaining a score for each resume, we can rank them in order of how closely they match the job description, enabling us to identify the most suitable candidates for the position. This score will serve as a key metric in our evaluation process, helping us to determine the effectiveness of our resume screening model.

Results and Insights

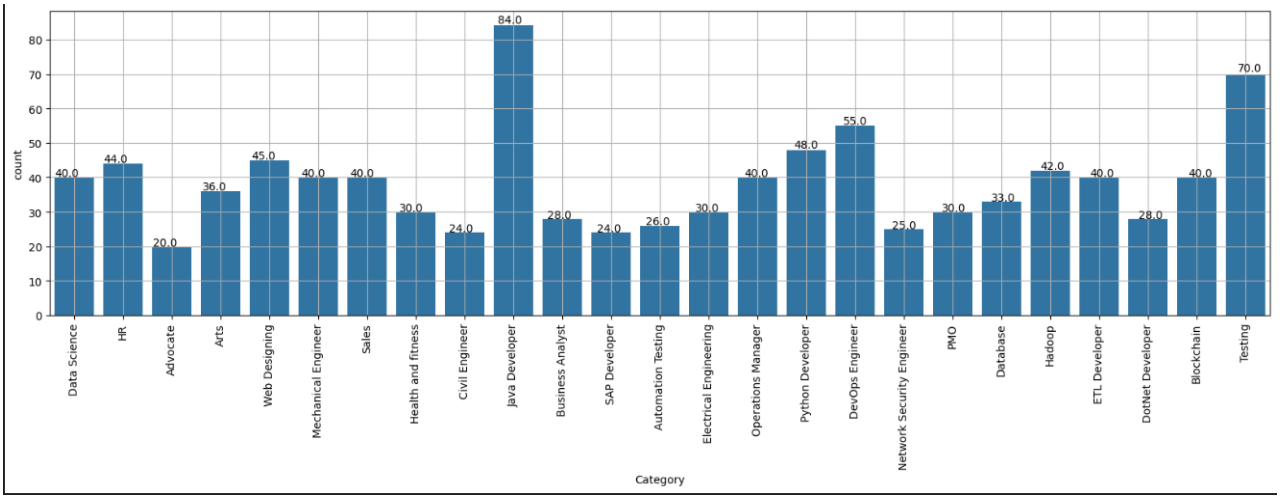

Figure 1 – Category distributions of Resumes

The histogram above displays the distribution of resume characteristics in our dataset, providing a visual representation of the data.

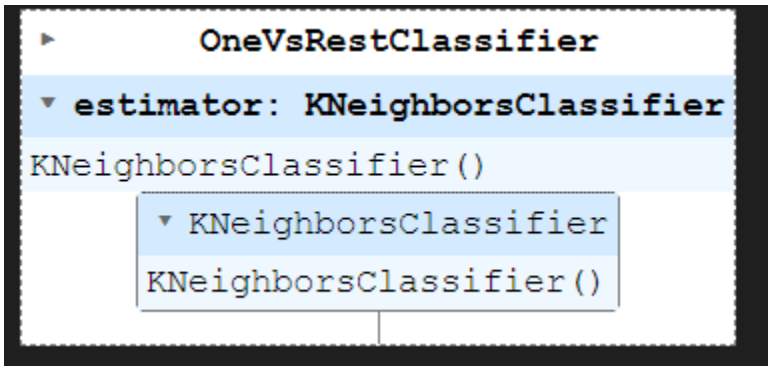

Figure 2: K-nearest neighbors Algorithm

The figure above illustrates the K-Nearest Neighbors (KNN) model trained on our resume dataset, which enables us to predict the similarity between resumes and identify top matches for a given job description. This model is built to classify resumes based on their characteristics and recommend the most suitable candidates.

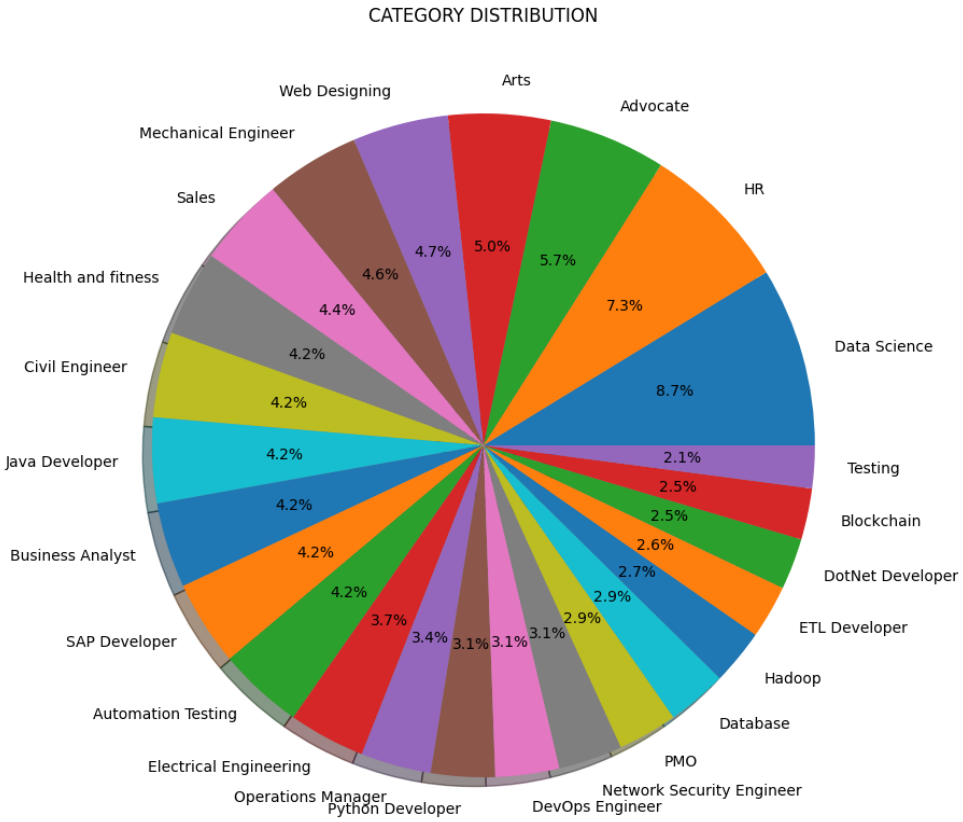

Figure 3: Pie Chart of resume distribution.

The pie chart above illustrates the proportion of different categories in our resume's dataset, providing a snapshot of the data distribution.

```
Resume 1 and Job Description similarity: 0.886466465096174
Resume 2 and Job Description similarity: 0.9598693627975485
Resume 3 and Job Description similarity: 0.9373529986209265
Resume 4 and Job Description similarity: 0.9304287748921176
Resume 5 and Job Description similarity: 0.9261109868402054
Resume 6 and Job Description similarity: 0.9612701665379259
Resume 7 and Job Description similarity: 0.9040944205165665
Resume 8 and Job Description similarity: 0.9545667170869343
Resume 9 and Job Description similarity: 0.9570088148082956
Resume 10 and Job Description similarity: 0.9364652893423902
Resume 11 and Job Description similarity: 0.9648733376050865
Resume 12 and Job Description similarity: 0.9154935457740526
Resume 13 and Job Description similarity: 0.8624301737201348
Resume 14 and Job Description similarity: 0.8567444189367546
Resume 15 and Job Description similarity: 0.8653103959557913
Resume 16 and Job Description similarity: 0.8572311875241316
Resume 17 and Job Description similarity: 0.8566464663912493
Resume 18 and Job Description similarity: 0.8765718915560775
```

Figure 4: Cosine Similarity index

The figure above displays the cosine similarity index for 18 resumes uploaded to the system, compared to a target job description. This index measures the similarity between each resume and the desired qualifications, enabling us to rank and categorize the profiles based on how well they match the job requirements. This allows HR to efficiently identify top candidates and streamline the recruitment process.

```
print('Accuracy of KNeighbors Classifier on training set: {:.2f}'.format(classifier.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set:     {:.2f}'.format(classifier.score(X_test, y_test)))
print(predictions)
```

✓ 2m 7.8s

```
Accuracy of KNeighbors Classifier on training set: 0.99
Accuracy of KNeighbors Classifier on test set:     0.98
[10 14  0 11 18 19 15  5 18 18 12  6 15  8  3 10 16 23  6 15 20 19 12 21
  8  6  7  6 17 12  1 21 22  5 22 17 10 12 15 15 22  5  1  4 23 24  8  6
 15 15 11 23 19 14  8 16  8 23  9  4 12 15  6 15  8  3 18 24  2 10 23  2
 22 13  0 15 19  2 13  0 20 14  1 16 21  9 23 20 23 17 18 24 10 13 20  1
 10  8 11  7  7 14 24 22 13 15  6  9 14  3  4 15 20  4 11 15 16 15  0 13
 15 19  6 10 20  3 13 12  8 11 24 16 11  6 21 18 18 14  5  7  1  5 13 15
 12 20 23  3 20 24 18 23 12 17 15  9  1 12 16  3 20 23  7 20 22 16 23 24
 23 17  7 23 11  1  8 13 19 23  8 10  4 24  3  2  3  4  9 22 24 21 23 22
 16]
```

Figure 5: K-nearest neighbors algorithm result

The figure above displays the results of the K-Nearest Neighbors (KNN) algorithm, showcasing the top matching resumes for a given job description. The algorithm's output highlights the most similar resumes, ranked by their proximity to the ideal candidate profile.

```
Classification report for classifier OneVsRestClassifier(estimator=KNeighborsClassifier()):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         7
           2       0.75      0.60      0.67         5
           3       1.00      1.00      1.00         8
           4       1.00      1.00      1.00         6
           5       1.00      1.00      1.00         5
           6       0.89      1.00      0.94         8
           7       1.00      0.86      0.92         7
           8       1.00      0.91      0.95        11
           9       1.00      1.00      1.00         5
          10       1.00      1.00      1.00         8
          11       0.86      1.00      0.92         6
          12       1.00      1.00      1.00         9
          13       1.00      1.00      1.00         8
          14       1.00      1.00      1.00         6
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00         8
          17       1.00      1.00      1.00         5
          18       1.00      1.00      1.00         8
          19       1.00      1.00      1.00         6
          20       1.00      1.00      1.00        10
...
   macro avg       0.98      0.97      0.97       193
weighted avg       0.98      0.98      0.98       193
```

Figure 5: Evaluation of k-nearest neighbor

The figure above displays the evaluation metrics for the K-Nearest Neighbors (KNN) model, assessing its performance in matching resumes with job descriptions. The evaluation metrics provide insight into the model's accuracy, precision, recall, and F1 score, helping us understand its strengths and areas for improvement.

Conclusion

In conclusion, this project results show that the effectiveness of the resume parser in helping both the efficiency and accuracy of screening process. By using Spacy for Natural Processing Language, Cosine Similarity for resume ranking and K-Nearest Neighbors for candidate matching, our application outperforms the traditional way of resume screening.

Moving forward, our main aim is to build a web application that can be user friendly. We also plan to incorporate additional customizable features and filters. Overall, this approach is a promising opportunity to transform the manual screening to a more automated task.

References

[1]. https://www.irjmets.com/uploadedfiles/paper//issue_4_april_2023/37428/final/fin_irjmets1683342426.pdf
[2] https://www.researchgate.net/publication/361772014_RESUME_PARSER
[3]. https://arxiv.org/abs/1301.3781