A WEB BASED APPLICATION FOR RESUME PARSER USING NATURAL LANGUAGE PROCESSING TECHNIQUES

Kanchan A Purohit[1], Anagha V[2], Chaitra Hasyagar[3] , Deeksha R[4], Deesha B Raj[5]

[1]Assistant Professor, Computer Science & Engineering, Bangalore Institute of Technology, Karnataka, India
[2,3,4,5] B.E Student, Computer Science & Engineering, Bangalore Institute of Technology, Karnataka, India

---

Abstract - *Resume is a powerful tool in a job search which outlines one's knowledge, skills, experience, expertise and accomplishments. An excellent resume attracts potential companies. It is very hard to find the right talent for a particular job. In order to make recruiters job easy, we have proposed a model which extracts details like education, projects, experience, phone no., email ID etc. and ranks the resumes based on company's specifications using Natural Language Processing (NLP) techniques. We have built a job portal where employees and applicants can upload resumes for a particular job, the required information will be parsed, structured resumes will be generated and the resumes of the employees will be ranked based on the company's skill set and employees skills mentioned in the resume.*

Key Words: **Parsing, Ranking, Natural Language Processing, K Nearest Neighbor, Vectorization, Cosine Similarity**

1. INTRODUCTION

Earlier recruiters used to spend most of their time in manually screening resumes. In this process, organizations would lose out on quality candidates and recruiters waste their time and effort. To avoid this, recruiters use automated Resume Parsing Systems in order to cut down on monotonous tasks and make recruitment easy. Resume Parsing System, lets recruiters find highly qualified candidates for a certain position. We have designed a model in which a parser will extract certain keywords like education, skills, experience, name, phone no., email ID etc. from the resume that is fed to it and ranks it accordingly for making the job of HR manager easier to select candidates for recruitment process.

A set of resumes from employees in .pdf format is taken as input and the employer gets a list of candidates that has been ranked according to parser and classifier. Applicants receive mail regarding status identified with their resumes.

In this proposed methodology, we are using Natural Language Processing technique for parsing the resume according to the particular companies. We use Named Entity recognizer (NER) which labels the data according to the type of their attribute and displays the text with the label. One of the platforms of NER is spaCy, an open source named entity visualizer that is used for information extraction. KNN algorithm is used for classification.

2. SYSTEM ARCHITECTURE

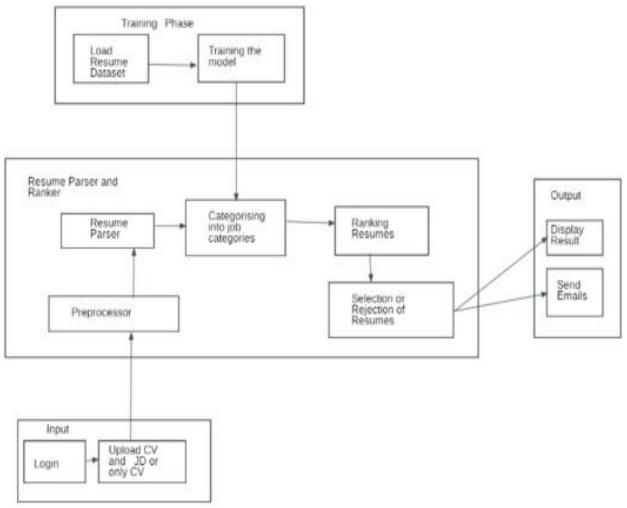

Fig -1: Architecture of the proposed system

- Training module: Loading dataset is a part of the training phase where we load our dataset to the model to test its accuracy. Training of the model is the next part of this phase where we train the model against the given dataset.
- Input: The candidate and recruiter upload the resumes in the required format.
- Processing: Each resume is parsed and the resumes are ranked according to their category.
- Output: The resumes along with their ranks are displayed and an option to send an email is given.

2.1 MODULE DESCRIPTION

- Login: In this component job seekers should register with our application and then they can login. Also in this component the recruiter can register or login to our application.

- Input Job Availability and Resume Uploader: The candidates upload their resume in pdf format. The recruiters can upload single or multiple resumes and can add the vacancy details and required skills.

- Resume Preprocessor: In this component, we remove any unnecessary information from resumes like stop words, hashtags, and special characters.

- Resume Parser: We are extracting the resume contents like name, email, education, skills, etc using the spacy parser function.

- Loading the dataset: In this component we load the labeled dataset into the system for testing against the untrained data.

- Building KNN classifier: In this component we use sklearn class to train a model to predict the suitable job title for their skills which is extracted from the seekers resume.

- Categorizing into jobs categories: Uses the trained model from the previous component to categorize each resume into corresponding job category.

- Resume Ranker: In this component system will compare the cosine similarity between the job seekers skills and recruiter's requirement skills for the corresponding job using NLP. Based on the similarity score resumes are ranked.

- Resume selection and rejection: Depending on the number of vacancies the resumes will get selected or rejected.

- Results are displayed: The list of ranked resumes is displayed to the recruiter.

- Emails are sent: Status emails regarding selection or rejection of their application are sent.

3. METHODOLOGY

The user registers or login to the system and uploads the resume.The system preprocesses , categorizes and ranks the resume and displays the output. The steps performed by the system are explained in the further sections.

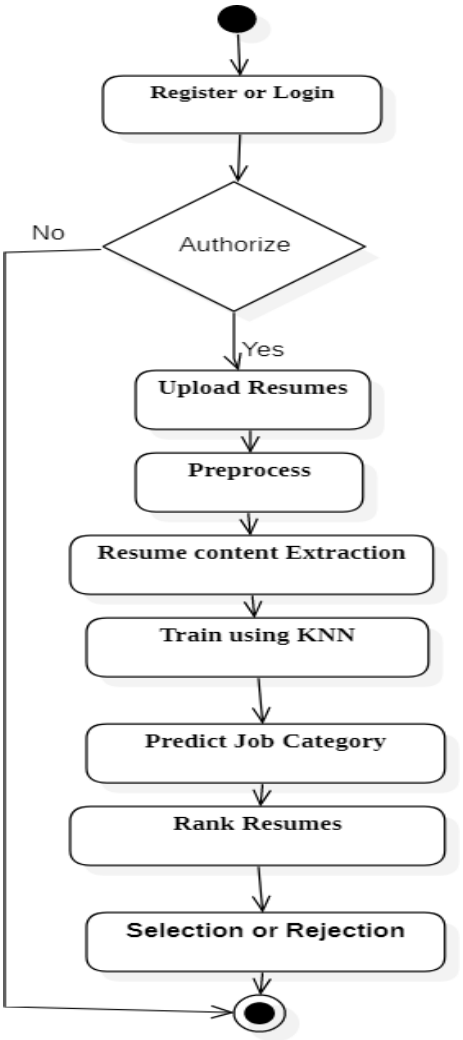

Fig -2: Activity diagram of the resume ranker application

3.1 PREPROCESSING

The natural language expressed in the resumes are not suitable for application of any machine learning or Natural language processing techniques. Hence the primary purpose of this preprocessing step is data cleaning. We have used spacy, an open-source natural language processing library to perform preprocessing of the resumes.

The first step in preprocessing is sentence detection. Each sentence is identified and isolated. The next step in preprocessing is tokenization. Tokenization is extracting the words and reducing them into their basic form. Example: extracting "organization" and reducing it into "organize". The reduced form of the word is called as a token.

In the next step, the punctuations and stop words are removed. Stop words are frequently used words that do

not increase the meaning of the sentence but make the sentence grammatically legible. Examples of stop words are: are, to, for, is, as etc.

In the next step, each token is reduced to a lemma. Lemma is the shortest form of a word that makes complete sense. It is the simplest form of the word that has a meaning. Example of reducing a token to a lemma is reducing "was" to "be".

Pattern matching is the next step in preprocessing. It is a natural language processing technique that matches phrase and tokens with a text corpus. Regular expressions are used to perform this task. Example of pattern matching is : for matching phone number the following regular expression is used: (r'[\+\(]?[1-9][0-9 .\-\(\)]{8,}[0-9]').Similarly the skills and required details are extracted from the resume.

3.2 TRAINING USING KNN ALGORITHM

K nearest neighbors is a lazy learner algorithm. The symbolic characteristic feature of a lazy learner algorithm is that it is trained during test time. KNN is a supervised learning algorithm. The input to the learning is a label set of input. We have used the KNN algorithm to classify the resumes in to suitable job categories. Once the category is defined for each of the resumes, the resumes can be ranked.

In the KNN algorithm implementation, the Euclidean distance is calculated between the instance, which has to be classified, and each of the instance in the training set is calculated. "k" is a predefined integer that is considered to identify the classification of the instance. Once the distances are computed, the "k" shortest distances are considered for computing the output of the instance. "k" is usually an odd integer to avoid conflicts. The label in majority of the "k" labels is assigned to the instance as the output.

Epoch is the number of times the instance is input to the training algorithm. Higher epoch results in output with high accuracy. We have defined the epoch to be 20 and the "k" value to be 5.

3.3 VECTORIZATION AND COSINE SIMILARITY

Vectorization is jargon for a classic approach of converting a computer file from its raw format (i.e. text) into vectors of real numbers which is the format that ML models support. This approach has been there ever since computers were first built, it's worked wonderfully across various domains, and it's now utilized in NLP. In Machine Learning, vectorization may be a step in feature extraction. The thought is to induce some distinct features out of the text for the model to coach on, by converting text to numerical vectors. There are many ways to perform vectorization, as we'll see shortly. We are going to see the one that's utilized in this particular project more closely which is TF-IDF. TF-IDF or Term Frequency–Inverse Document Frequency, may be a numerical statistic that's intended to reflect how important a word is to a document. In Bag of Words, the vectorization is simply concerned with the frequency of vocabulary words in an exceedingly given document. As a result, articles, prepositions, and conjunctions which don't contribute much to the meaning get the maximum amount of importance as, say, adjectives. TF-IDF helps us to beat this issue. Words that get repeated too often don't overpower less frequent but important words. It has two parts:

i) TF

TF stands for Term Frequency. So one can imagine that this number will always stay ≤ 1, thus we now judge how frequent a word is within the context of all of the words in a document. We calculate this using the following formula:

$$TF = \frac{Frequency\ of\ word\ in\ a\ document}{Total\ number\ of\ words\ in\ that\ document}$$

ii) IDF

It stands for Inverse Document Frequency, but before we get into IDF, we must be aware of DF – Document Frequency. DF tells us about the proportion of documents that contain a particular word. IDF is the reciprocal of the Document Frequency. The intuition behind it's that the more common a word is across all documents, the lesser its importance is for this document. A logarithm is taken to dampen the effect of IDF within the final calculation. We use the following formula to calculate DF:

$$DF = \frac{Documents\ containing\ word\ W}{Total\ number\ of\ documents}$$

We use the following formula to calculate IDF:

$$IDF = \log(\frac{Total\ number\ of\ documents}{Documents\ containing\ word\ W})$$

This can be how TF-IDF manages to include the importance of a word. The higher the score, the more important that word is. Despite being so simple, TF-IDF is understood to be extensively utilized in tasks like Information Retrieval to gauge which response is the best for a question , especially useful during a chatbot or in Keyword Extraction to work out which word is the most relevant in a document.

Cosine similarity may be a metric accustomed to measure how similar the documents are no matter their size. Mathematically, it measures the cosine of the angle between two vectors projected in a very multidimensional space. The cosine similarity is advantageous because whether or not the 2 similar documents are far apart by the Euclidean distance (due to the dimensions of the document), likelihood is that they'll still be oriented closer together. The smaller the angle, higher the cosine similarity. It does this by calculating the similarity score between the vectors, which is finished by finding the angles between them. The range of similarities is between 0 and 1. If the worth of the similarity score between two vectors is 1, it means there's a greater similarity between the 2 vectors.

On the opposite hand, if the worth of the similarity score between two vectors is 0, it means there's no similarity between the 2 vectors. When the similarity score is one, the angle between two vectors is 0 and when the similarity score is 0, the angle between two vectors is 90 degrees. Cosine Similarity is one in all the methods to seek out similarities between the 2 documents.

3.4 RESULTS

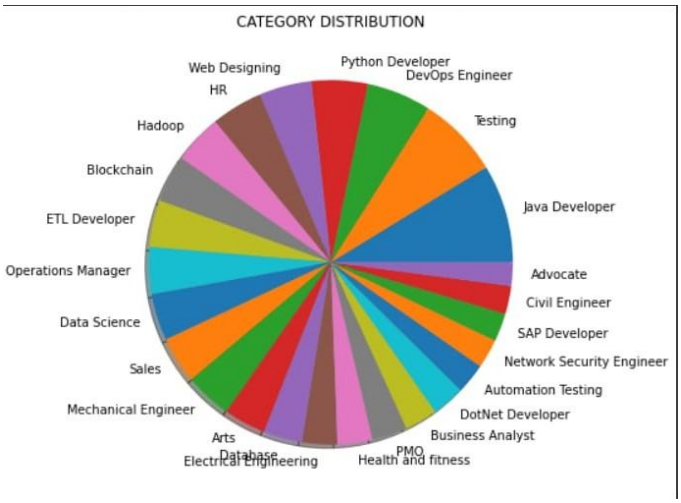

Fig -3: Job category distribution pie chart

We are training the model using KNN classification model. We calculate the accuracy by dividing the amount of correct predictions by the entire number of samples. Among the 24 classes available we calculate the precision, recall, f1 score and support to know whether the training set is being classified properly. Precision is defined because the ratio of correctly classified positive samples (True Positive) to a complete number of classified positive samples. The recall is calculated because the ratio between the numbers of Positive samples correctly classified as Positive to the full number of Positive samples. Support is the total entries of every class within the actual dataset. The ultimate accuracy recorded at the top of the report is 0.99.

4. CONCLUSIONS

Our approach is to make the work of companies and candidates easier and effective. The aim of the proposed model is to ease the recruitment process. The process ensures to provide quality applicants to the companies. Unfair and discriminatory practises will be dampened. Resumes are ranked on the basis of information provided in the form of technical skills. However, the proposed model has a few limitations. This process requires resumes to be in a particular format. The recruiter can add the requirements of only one job category at a time after which a list of ranked resumes for the set of resumes that were uploaded can be viewed. If there is missing data within a resume, the respective candidate is not notified and the resume gets rejected.

REFERENCES

[1]. Ayishathahira and Sreejith, "Combination of neural networks and conditional random fields for efficient resume parsing", International CET Conference on Control, Communication and Computing(IC4), 2018.

[2]. Dr.Parkavi A,Pooja Pandey,Poornima J,Vaibhavi G S,Kaveri BW, "E-Recruitment system through resume parsing, psychometric test and social media analysis", IJARBEST, 2019.

[3]. Papiya Das, Manjusha Pandey, Siddharth Swarup Rautaray, "A CV parser model using entity extraction process and big data tools ", IJITCS, 2018.

4]. Nirali Bhaliya, Jay Gandhi, Dheeraj Kumar Singh, "NLP based extraction of relevant resume using machine learning ", IJITEE, 2020.

[5]. Fahad, SK Ahammad, and Abdulsamad Ebrahim Yahya, "Inflectional review of deep learning on natural language processing", International Conference on Smart Computing and Electronic Enterprise (ICSCEE), 2018.

[6]. Shicheng Zu and Xiulai Wang, "Resume information extraction with a novel textblock segmentation algorithm", International Journal on Natural Language Computing (IJNLC) Vol.8, No.5, October 2019.

[7]. Y. Deng, H. Lei, X. Li and Y. Lin, "An improved deep neural network model for job matching", International Conference on Artificial Intelligence and Big Data (ICAIBD) 2018.

[8]. S. Sawleshwarkar, N.Rangnani, V. Mariwalla and A.Halbe, "Simplified recruitment model using Text-Mining on psychometric and aptitude test", Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) 2018.

[9]. H. Suen and H. Chen, "Screening passive job seekers on facebook", International Cognitive Cities Conference (IC3) 2018.

[10]. Mujtaba, Dena F., and Nihar R. Mahapatra, "Ethical considerations in AI-based recruitment", IEEE International Symposium on Technology and Society (ISTAS) 2019.