RESUME PARSING USING NLP

[1]Mrs N.Anjamma, [2]N.Revanth Kumar, [3]N. Vishnusree, [4]P. Sai Manasa

[1]Assistant Professor, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,

[2]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad, revanth02324@gmail.com

[3]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad, Vishnusreenuligonda@gmail.com

[4]BTech student, Dept.of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad, manasaperumandla1@gmail.com

Abstract: *The resume parser helps us to convert the amorphous form of resume data into a structured format. A resume parser analyses resume data and extract the most important information that automatically stores, organizes, and analyses to find the best candidates. For this project, we collected the data from 25 different category resumes for developing a Machine learning model to find which category the resume belongs to. We build a resume parser to extract text from two different files docx and pdf files. Using the Python-docx library we extract the text when the resume is in a docx file and if the resume is in pdf format means we use the PyMuPDF library for extracting the text. For converting the text into numerical data, we implemented term frequency and inverse document frequency concept which gives importance to each word based on probability. The ML models we developed were used to find the category of the resume to which it belongs ex: java developer, data scientist, Full stack developer. Email address and Phone number are well-defined patterns in themselves. Thus, we would be using Regular Expressions in order to capture them in the resume. So, with the help of ML models and python, we developed a resume parser application that can extract the important information from the resume.*

Keywords: *Resume Parsing, Machine learning, Natural language processing*

I. INTRODUCTION

Resumes are commonly presented in PDF or MS word format, and there is no particular structured format to present/create a resume. So, we can say that each individual would have created a different structure while preparing their resumes. It is easy for us human beings to read and understand those unstructured or rather differently structured data because of our experiences and understanding, but

ISSN: 2366-1313

machines don’t work that way. Machines cannot interpret it as easily as we can. The conversion of cv/resume into formatted text or structured information to make it easy for review, analysis, and understanding is an essential requirement where we have to deal with lots of data. Basically, taking an unstructured resume/cv as an input and providing structured output information is known as resume parsing.

Subsequently, converting a resume into prepared text or structured information makes studying, analysing, and comprehending easier. As a result, many organizations and institutions depend on Information Extraction, were unstructured data and vital information

are extracted and converted to make information more readable and organized data forms.

The completion of this task takes a long time for humans. So, it is necessary to develop an

automated intelligent system that can extract all relevant information to determine whether

an applicant is suitable for a particular job profile.

Resume parser is an NLP model that can extract information like Skill, University, Degree, Name, Phone, Designation, Email, other social media links, Nationality, etc. irrespective of their structure. To create such an NLP model that can extract various information from resume, we have to train it on a proper dataset. And we all know, creating a dataset is difficult if we go for manual tagging. To reduce the required time for creating a dataset, we have used various techniques and libraries in python, which helped us identifying required information from resume.

Email address and Phone number are well-defined patterns in themselves. Thus, we would be using Regular Expressions in order to capture them in the resume. This may seem easy but in reality, one of the most challenging tasks of resume parsing is to extract the person’s name. There are millions of names around the world and living in a globalized world, we may come up with a resume from anywhere. For extracting the skills, we collected the top 50 skills and using spacy matcher we mapped each word from the text with 50 skills so the words which match the condition can belong to the skills category. This technique helps us to find the skills from the resume.

Corporate companies and recruitment agencies process numerous resumes daily. This is no task for humans. An automated intelligent system is required which can take out all the vital information from the unstructured resumes and transform all of them to a common structured format which can then be ranked for a specific job position. Parsed information include name, email address, social profiles, personal websites, years of work experience, work experiences, years of education, education experiences, publications, certifications, volunteer experiences, keywords and finally the cluster of the resume (ex: computer science, human resource, etc.). The parsed information is then stored in a database (NoSQL in this case) for later use. Unlike other unstructured data (ex: email body, web page contents, etc.), resumes are a bit structured. Information is stored in discrete sets. Each set contains data about the person's contact, work experience or education details. In spite of this resumes are difficult to parse. This is because they vary in types of information, their order, writing style, etc. Moreover, they can be written in various formats. Some of the common ones include '.txt', '.pdf', '.doc', '.docx', '.odt', '.rtf' etc. To parse the data from different kinds of resumes effectively and efficiently, the model must not rely on the order or type of data.

II. LITERATURE SURVEY

Agencies and different high-level companies have to deal with an extreme number of new jobs seeking employees with different resumes. However, looking after those large numbers of text data and filtering out the needed candidates is a burden on the brain and more time consuming. Therefore, the essence of this literature review is on studying resumes in different formats such as single-column resumes, double-column resumes with extension.pdf,.docx, and how the suggested Information Extraction System converts that unstructured information into structured layout through Parsing. Accordingly, this review also helps to understand and apply several in-use and well recognized algorithms currently being used in industries to reduce human labor. Depending upon the Company's preference to hire employees, the Extraction System will manage the gathered information with more readability and organized data forms. Furthermore, the analysis of various machine learning algorithms and natural language processing techniques would be equally carried out along with their proper implementation and evaluation. The reviews from multiple research publications and journals are included below.

The use of two natural language processing algorithms to extract important data from an unstructured multilingual CV has provided a solution for selecting relevant document parts and the similar particular information at the low hierarchy level (VUKADIN, et al., 2021). It uses a machine learning method in NLP to obtain a high level of extraction accuracy. The authors claim that they have solved the CV parsing challenge by building an NLP system. The recently introduced tokens [NEW LINE] and [SKILL] are shown to have trained to perform as expected.

Recruitment has evolved rapidly in the last decade, from traditional job fairs to webbased
recruitment platforms. As a result, (Wang & Zu, 2019) presented a resume parsing
pipeline that uses neural network-based classifiers and distributed embeddings from
beginning to end. The pipeline eliminates the laborious process of manually creating several
handcrafted elements.

(Kopparapu, 2015)proposes a natural language processing (NLP) system that focuses on automated information extraction from resume to facilitate speedy resume search and management for structured and unstructured resumes. The reviewed literature addressed a wide range of information extraction concepts and research-based procedures and their main approaches. The majority of the research material uses a combination of ML and DLbased methods. After analyzing all of the review findings, it is easier to conclude that extracting meaningful information from resumes using NLP and its different techniques like Regex and Spacy simplifies the recruiting process significantly minimizes time complexity in a larger way.

HISTORY OF HIRING

The process of hiring has evolved over the period of time. In the first-generation hiring model, the companies would advertise their vacancies on newspapers and television. The applicants would send in their resumes via post and their resumes would be sorted manually. Once shortlisted, the hiring team would call the applicants for further rounds of interview. Needless to say, this was a time-consuming procedure. But the industries started growing and so did the hiring needs. Hence the companies started outsourcing their hiring process. Hiring consultancies came into existence. These agencies required the applicants to upload their resumes on their websites in particular formats. The agencies would then go through the structured data and shortlist candidates for the company. This process had a major drawback. There were numerous agencies and each had their own unique format. To overcome all the above problems an intelligent algorithm was required which could parse information from any unstructured resumes, sort it based on the clusters and rank it finally. The model uses natural language processing to understand the resume and then parse the information from it. Once information is parsed it is stored in the database. When the employer posts a job opening, the system ranks the resumes based on keyword matching and shows the most relevant ones to the employer.

III. PROPOSED METHODOLOGY

The main objective of Natural Language Processing (NLP)-based Resume Parser using nlp project is to extract the required information about candidates without having to go through each and every resume manually, which ultimately leads to a more time and energy efficient process. Resumes are commonly presented in PDF or MS word format, and there is no particular structured format to present a resume. So, by using this project we can convert the resume to a structured format.

A purpose system can help in resolving the challenge of obtaining useful information from a resume in a structured format. By resolving this issue, recruiters will be able to save hours each day by eliminating manual resume screening. Bias in hiring is still prevalent, thus this method may also address the bias hiring process and strengthen a non-bias policy.

SYSTEM ARCHITECTURE

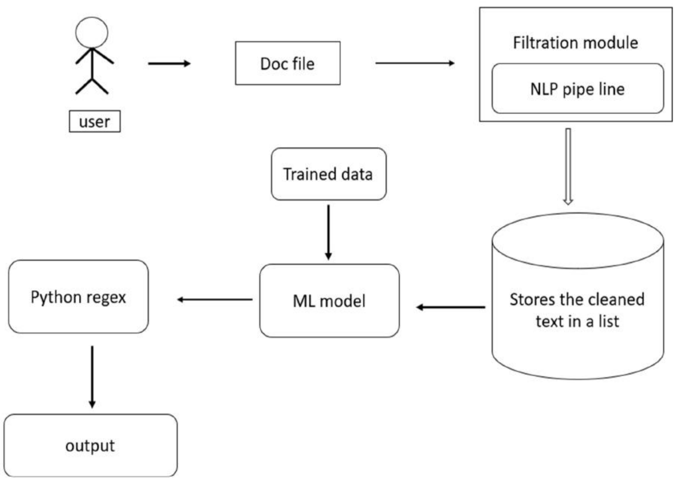
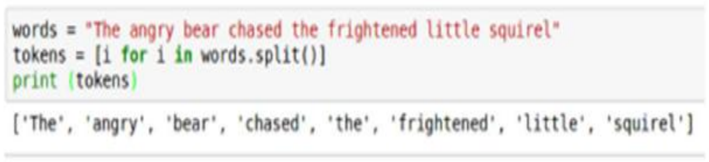


Fig.1 System architecture

A) NLP TOKENIZATION

Tokenization is the task of chopping off a provided character sequence and a detailed document unit. It does away with certain characters like punctuation and the chopped units are further called tokens. It can be illustrated as follows:



```
words = "The angry bear chased the frightened little squirel"
tokens = [i for i in words.split()]
print (tokens)
['The', 'angry', 'bear', 'chased', 'the', 'frightened', 'little', 'squirel']
```

Tokens are usually referred to as terms or words, but sometimes fabricating a type/token distinction is essential. A specimen of an array of characters in a document that is assembled as a helpful acceptable unit for processing is called a token. Whereas, the group of tokens which consists of same character sequence is called type. And a type that is added to the dictionary of IR system is called term. We can completely differentiate a set of index terms from tokens. As an example, we can say, they can be acceptable identifiers in taxonomy, but mostly in modern IR systems, they have a strong relation with tokens in the document. Nevertheless, as a substitute for being totally the tokens appearing in the document, they are mostly derived from them by various processes of normalization

B) POS TAGGING

we'll look at each token and try to guess it's part of speech whether it is a noun, a verb, an adjective and so on. Knowing the role of each word in the sentence will help us start to figure

out what the sentence is talking about.

We can do this by feeding each word (and some extra words around it for context) into a pretrained part-of-speech classification model:

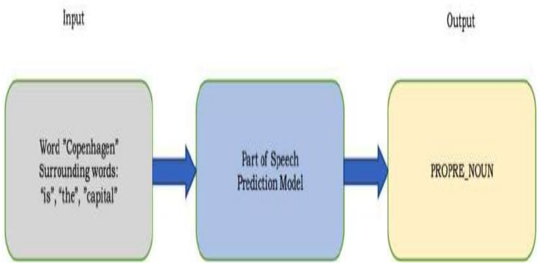
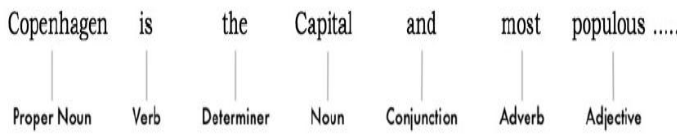

Input
Output
Word "Copenhagen" Surrounding words: "is", "the", "capital"
Part of Speech Prediction Model
PROPRE_NOUN

Fig.2 Parts of speech

The part-of-speech model was originally trained by feeding it millions of English sentences with each word's part of speech already tagged and having it learn to replicate that behaviour.

Keep in mind that the model is completely based on statistics — it doesn't actually understand what the words mean in the same way that humans do. It just knows how to guess a part of speech based on similar sentences and words it has seen before

After processing the whole sentence, we'll have a result like this:



C) STOP WORDS

we want to consider the importance of each word in the sentence. English has a lot of filler

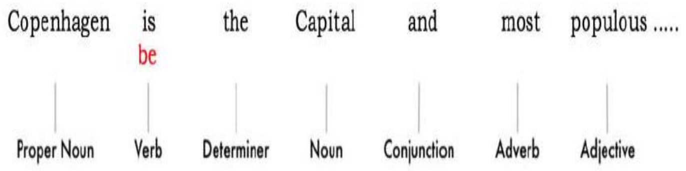
words that appear very frequently like "and", "the", and "a". When doing statistics on text,

these words introduce a lot of noise since they appear way more frequently than other words.

Some NLP pipelines will flag them as stop words — that is, words that you might want to

filter out before doing any statistical analysis.

Here's how our sentence looks with the stop words grayed out:



Stop words are usually identified by just by checking a hardcoded list of known stop words.

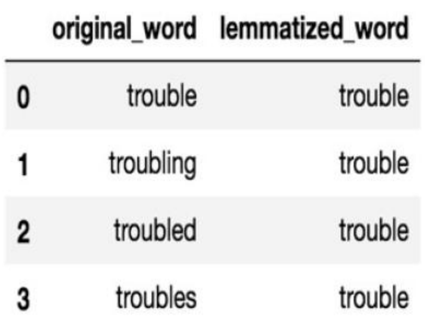
But there's no standard list of stop words that is appropriate for all applications. The list of
words to ignore can vary depending on your application.
For example, if you are building a rock band search engine, you want to make sure you don't
ignore the word "The". Because not only does the word "The" appear in a lot of band names,
there's a famous 1980's rock band called The The!

D) LEMMATIZATION

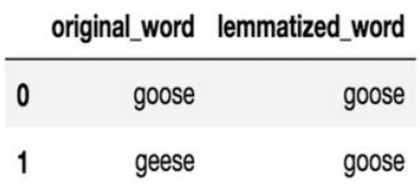
English is also one of the languages where we can use various forms of base words. When working on the computer, it can understand that these words are used for the same concepts when there are multiple words in the sentences having the same base words. The process is what we call lemmatization in NLP.

IV. RESULTS

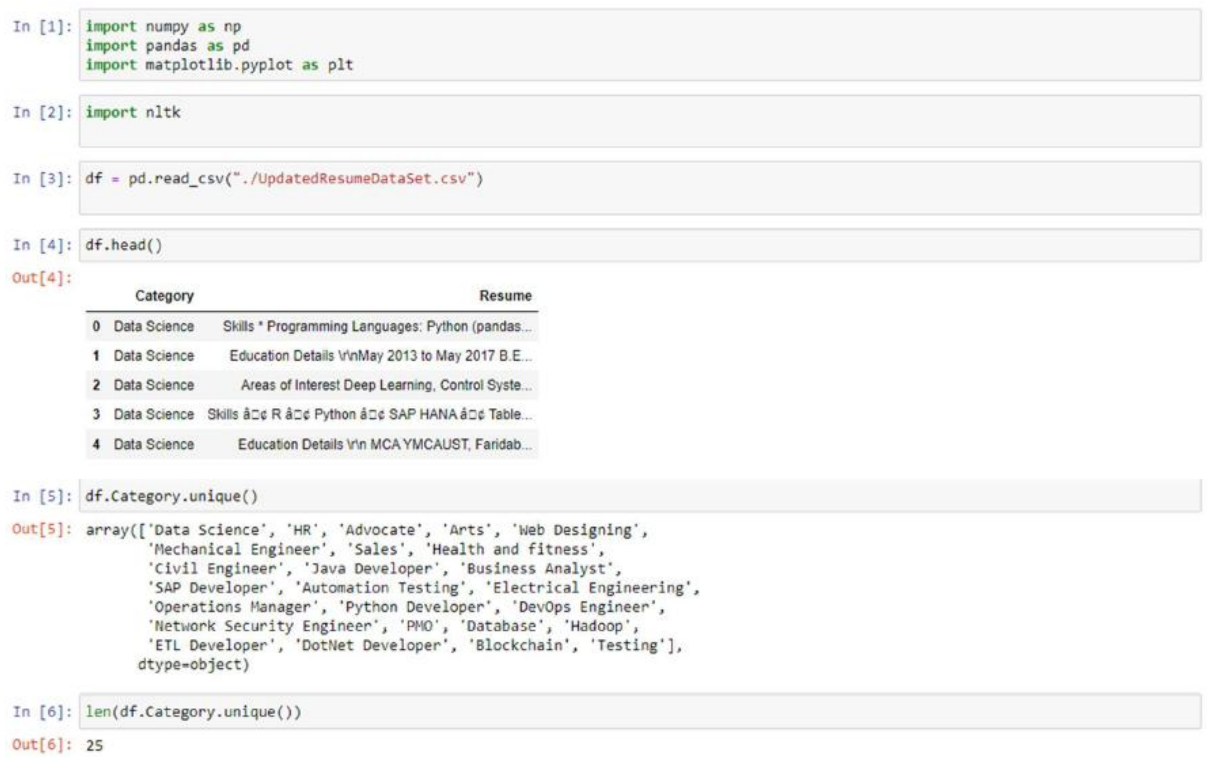
It goes to the root level to find out the base form of all the available words. They have ordinary rules to handle the words, and most of us are unaware of them. We can also lemmatize verbs by finding their root, unconjugated form. So "I had two MacBooks" becomes "I [have] two [macbook]."



	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble



	original_word	lemmatized_word
0	goose	goose
1	geese	goose

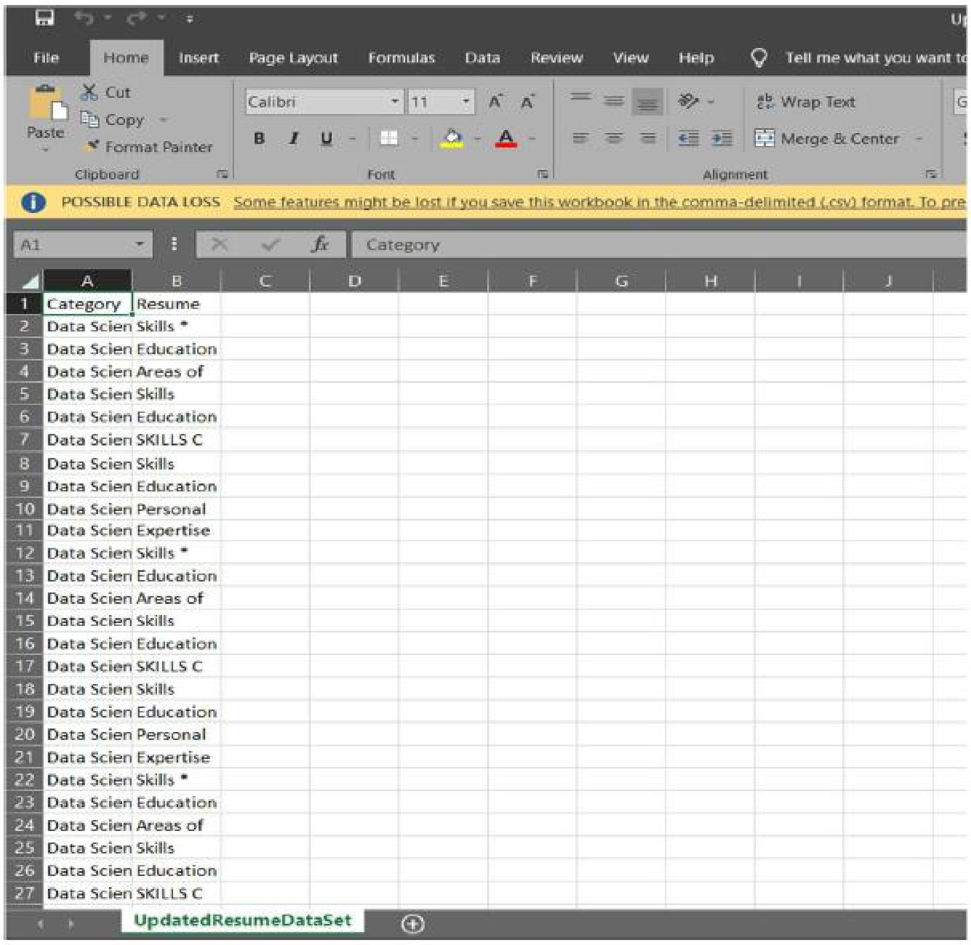


```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

In [2]: import nltk

In [3]: df = pd.read_csv("./UpdatedResumeDataSet.csv")

In [4]: df.head()
Out[4]:
```

	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â¢ R â¢ Python â¢ SAP HANA â¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...

```
In [5]: df.Category.unique()
Out[5]: array(['Data Science', 'HR', 'Advocate', 'Arts', 'Web Designing',
               'Mechanical Engineer', 'Sales', 'Health and fitness',
               'Civil Engineer', 'Java Developer', 'Business Analyst',
               'SAP Developer', 'Automation Testing', 'Electrical Engineering',
               'Operations Manager', 'Python Developer', 'DevOps Engineer',
               'Network Security Engineer', 'PMO', 'Database', 'Hadoop',
               'ETL Developer', 'DotNet Developer', 'Blockchain', 'Testing'],
              dtype=object)

In [6]: len(df.Category.unique())
Out[6]: 25
```

Fig.3 Jupyter Interface

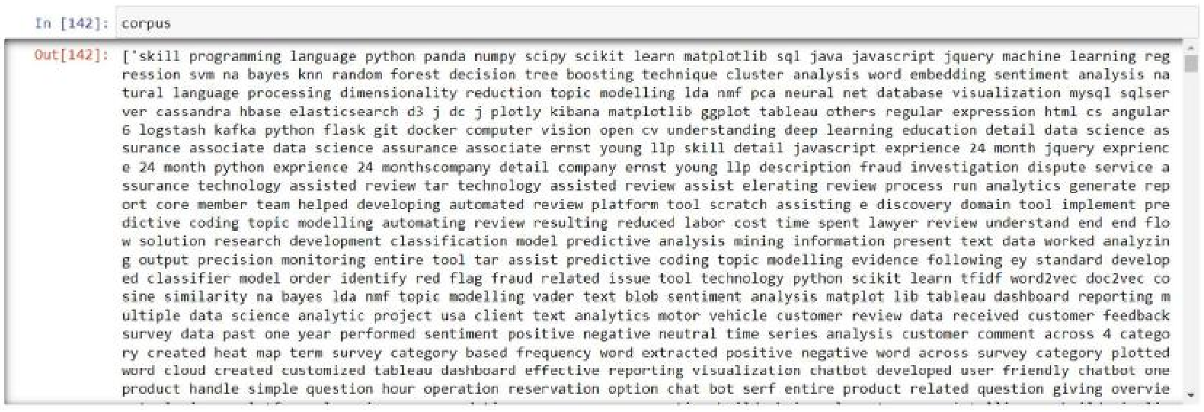


POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To pre

A1 Category

	A	B
1	Category	Resume
2	Data Scien	Skills *
3	Data Scien	Education
4	Data Scien	Areas of
5	Data Scien	Skills
6	Data Scien	Education
7	Data Scien	SKILLS C
8	Data Scien	Skills
9	Data Scien	Education
10	Data Scien	Personal
11	Data Scien	Expertise
12	Data Scien	Skills *
13	Data Scien	Education
14	Data Scien	Areas of
15	Data Scien	Skills
16	Data Scien	Education
17	Data Scien	SKILLS C
18	Data Scien	Skills
19	Data Scien	Education
20	Data Scien	Personal
21	Data Scien	Expertise
22	Data Scien	Skills *
23	Data Scien	Education
24	Data Scien	Areas of
25	Data Scien	Skills
26	Data Scien	Education
27	Data Scien	SKILLS C

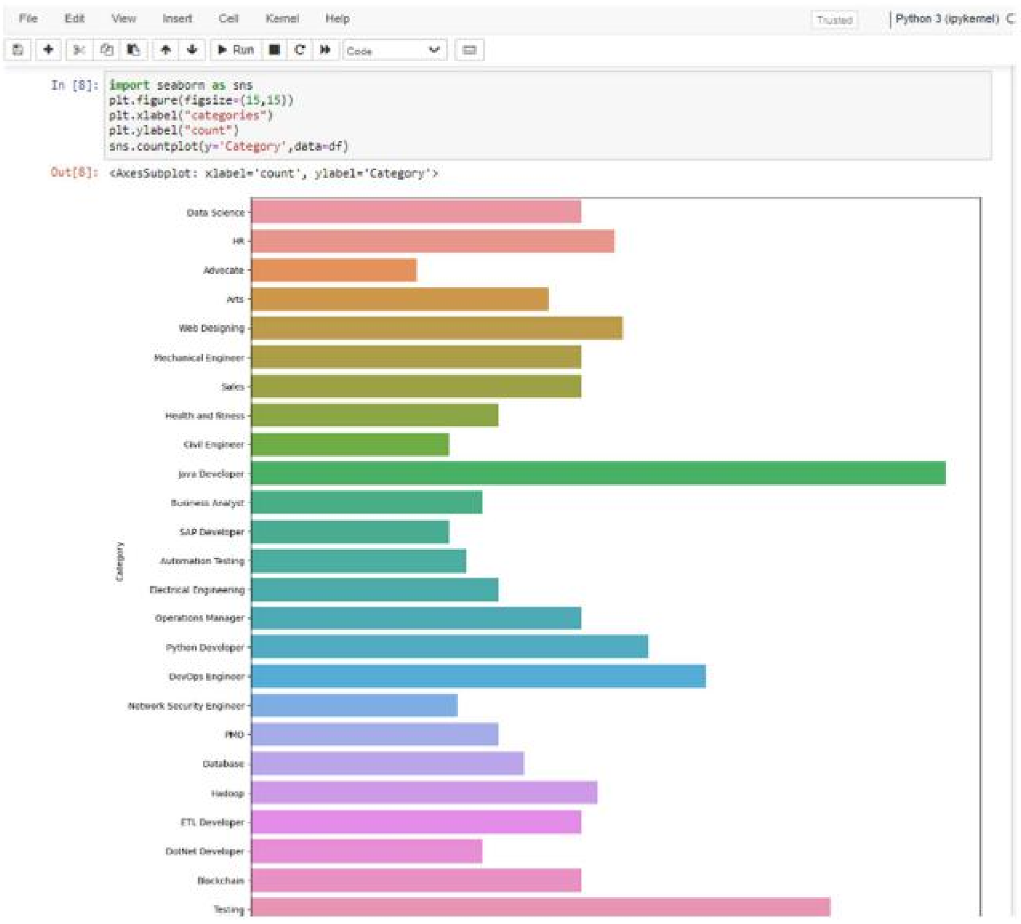
Fig.4 Resume Data set



```
In [142]: corpus
```

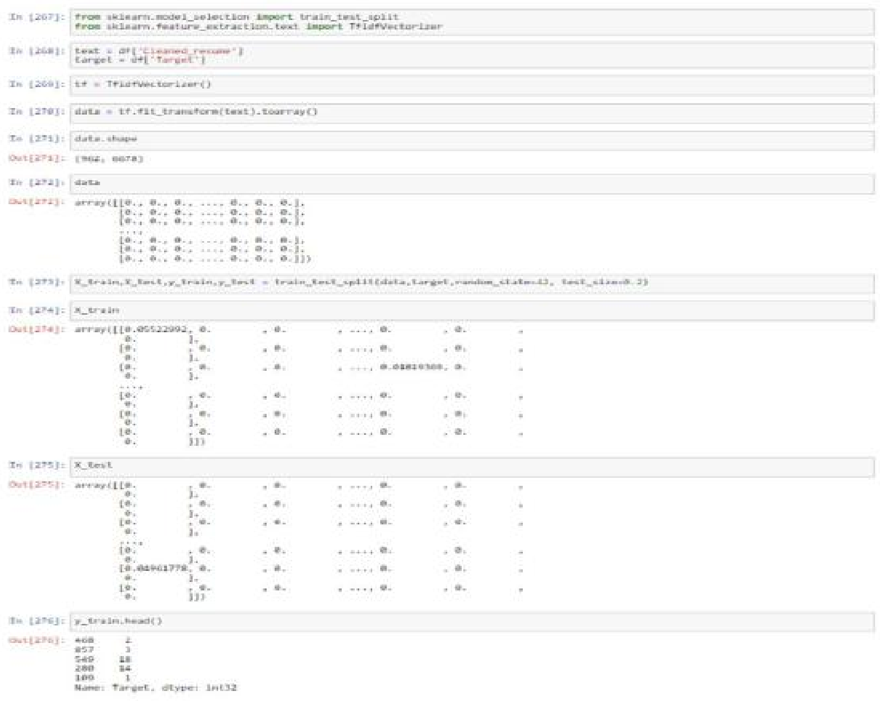
```
Out[142]: ['skill programming language python panda numpy scipy scikit learn matplotlib sql java javascript jquery machine learning reg
ression svm na bayes knn random forest decision tree boosting technique cluster analysis word embedding sentiment analysis na
tural language processing dimensionality reduction topic modelling lda nmf pca neural net database visualization mysql sqlser
ver cassandra hbase elasticsearch d3 j dc j plotly kibana matplotlib ggplot tableau others regular expression html cs angular
6 logstash kafka python flask git docker computer vision open cv understanding deep learning education detail data science as
surance associate data science assurance associate ernst young llp skill detail javascript exprience 24 month jquery exprienc
e 24 month python exprience 24 monthscompany detail company ernst young llp description fraud investigation dispute service a
ssurance technology assisted review tar technology assisted review assist elerating review process run analytics generate rep
ort core member team helped developing automated review platform tool scratch assisting e discovery domain tool implement pre
dictive coding topic modelling automating review resulting reduced labor cost time spent lawyer review understand end end flo
w solution research development classification model predictive analysis mining information present text data worked analyzin
g output precision monitoring entire tool tar assist predictive coding topic modelling evidence following ey standard develop
ed classifier model order identify red flag fraud related issue tool technology python scikit learn tfidf word2vec doc2vec co
sine similarity na bayes lda nmf topic modelling vader text blob sentiment analysis matplot lib tableau dashboard reporting m
ultiple data science analytic project usa client text analytics motor vehicle customer review data received customer feedback
survey data past one year performed sentiment positive negative neutral time series analysis customer comment across 4 catego
ry created heat map term survey category based frequency word extracted positive negative word across survey category plotted
word cloud created customized tableau dashboard effective reporting visualization chatbot developed user friendly chatbot one
product handle simple question hour operation reservation option chat bot serf entire product related question giving overvie
```

Fig.5 Corpus data



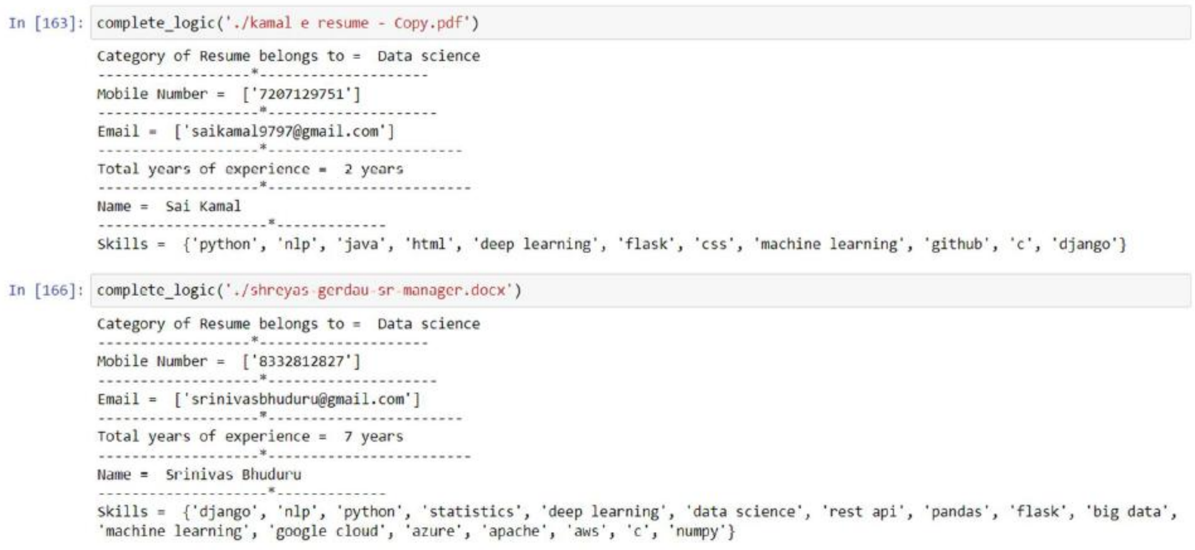
```
In [8]: import seaborn as sns
        plt.figure(figsize=(15,15))
        plt.xlabel("categories")
        plt.ylabel("count")
        sns.countplot(y='Category',data=df)

Out[8]: <AxesSubplot: xlabel='count', ylabel='Category'>
```

Fig.6 Graphical representation of categories from dataset



```
In [267]: from sklearn.model_selection import train_test_split
          from sklearn.feature_extraction.text import TfidfVectorizer

In [268]: text = df['Cleaned_resume']
          target = df['Target']

In [269]: tf = TfidfVectorizer()

In [270]: data = tf.fit_transform(text).toarray()

In [271]: data.shape
Out[271]: (962, 6678)

In [272]: data
Out[272]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])

In [273]: X_train,X_test,y_train,y_test = train_test_split(data,target,random_state=42, test_size=0.2)

In [274]: X_train
Out[274]: array([[0.05522992, 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.01819508, 0.        ,
                  0.        ],
                 ...,
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ]])

In [275]: X_test
Out[275]: array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 ...,
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.04961778, 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ]])

In [276]: y_train.head()
Out[276]: 468     2
          857     3
          549    18
          280    14
          100     1
          Name: Target, dtype: int32
```

Fig.7 Train Test Split



```
In [163]: complete_logic('./kamal e resume - Copy.pdf')
          Category of Resume belongs to =  Data science
          -----------------*-------------------
          Mobile Number =  ['7207129751']
          ------------------*-------------------
          Email =  ['saikamal9797@gmail.com']
          -------------------*-----------------------
          Total years of experience =  2 years
          -------------------*------------------------
          Name =  Sai Kamal
          --------------------*------------
          Skills =  {'python', 'nlp', 'java', 'html', 'deep learning', 'flask', 'css', 'machine learning', 'github', 'c', 'django'}

In [166]: complete_logic('./shreyas-gerdau-sr-manager.docx')
          Category of Resume belongs to =  Data science
          -----------------*-------------------
          Mobile Number =  ['8332812827']
          ------------------*-------------------
          Email =  ['srinivasbhuduru@gmail.com']
          -------------------*-----------------------
          Total years of experience =  7 years
          -------------------*------------------------
          Name =  Srinivas Bhuduru
          --------------------*------------
          Skills =  {'django', 'nlp', 'python', 'statistics', 'deep learning', 'data science', 'rest api', 'pandas', 'flask', 'big data',
          'machine learning', 'google cloud', 'azure', 'apache', 'aws', 'c', 'numpy'}
```

Fig.8 Parsed resume is generated

V. CONCLUSION

A normal resume is a compilation of information about a person's work experience, academic background, qualifications, and personal details. These elements might be present in a variety of ways or not at all. It's difficult to keep up with the jargon used in resumes. A resume is made up of corporate names, institutions, degrees, and other information that can be written in a variety of ways. It will take time to review all the resume by an individual.

Machine works faster than human and their accuracy to do any task was also good. Therefore, I have made a system which includes machine learning that extract the important information from resumes within a minute or less than a minute. The hiring individual can use this system for hiring any individual. From this project we conclude that with the help of ML models and python, we developed a resume parser application that can extract the important information from the resume. It is more constructive for companies to hire the candidates by selecting the main objects from their resumes which are derived by using this resume parsing application. Our approach is to make the work of companies and candidates easier and effective. Basically, our aim is to ease the recruitment process. The process will provide the quality of applicants for the companies. The unfair and discriminatory practice in the process will be dampened.

REFERENCES

[1]. F. Ciravegna, "Adaptive information extraction from text by rule induction and generalisation," in Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI2001), 2001.

[2]. A. Chandel, P. Nagesh, and S. Sarawagi, "Efficient batch top-k search for dictionary-based entity recognition," in Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE), 2006.

[3]. S. Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kauffman, 2002

[4]. M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni, "KnowItNow: Fast, scalable information extraction from the web," in Conference on Human Language Technologies (HLT/EMNLP), 2005.

[5]. M. J. Cafarella and O. Etzioni, "A search engine for natural language applications," in WWW, pp. 442–452, 2005.

[6]. https://www.ijircce.com/upload/2016 /april/218_ Intellig ent.pdf

[7]. https://www.tutorialspoint.com/compiler_ design/images /token_passing.jpg

[8]. http://www.nltk.org/book/tree_images/ch0 8-tree-6.png

[9] C, B. P., 2020. towardsai. [Online] Available at: https://towardsai.net/p/nlp/natural-language-processing-conceptsandworkflow- 48083d2e3ce7 [Accessed 2022].

[10] Bhatia, V., Rawat, P., Kumar, A. & Shah, R. R., 2019. End-to-End Resume Parsing and Finding Candidates for a Job Description using BERT. arxiv. • chavan, J., 2020. medium. [Online] Available at: https://medium.com/@jeevanchavan143/nlp tokenization stemming lemmatization-bag-of-words-tf-idf-pos-7650f83c60be [Accessed 2022].