Resume Classification using various Machine Learning Algorithms

Riya Pal[1,*], *Shahrukh* Shaikh[2], *Swaraj* Satpute[3] and *Sumedha* Bhagwat[4]

[1,2,3] Ramrao Adik Institute of Technology, Nerul, Navi Mumbai, India
[4] D.Y. Patil Deemed to be University, Ramrao Adik Institute of Technology, Nerul, Navi Mumbai, India

Abstract. With the onset of the epidemic, everything has gone online, and individuals have been compelled to work from home. There is a need to automate the hiring process in order to enhance efficiency and decrease manual labour that may be done electronically. If resume categorization were done online, it would significantly save paperwork and human error. The recruiting process has several steps, but the first is resume categorization and verification. Automating the first stage would greatly assist the interview process in terms of speedy applicant selection. Classification of resumes will be performed using Machine Learning Algorithms such as Nave Bayes, Random Forest, and SVM, which will aid in the extraction of skills and show diverse capabilities under appropriate job profile classes. While the abilities are being extracted, an appropriate job profile may be retrieved from the categorised and pre-processed data and shown on the interviewer's screen. During video interviews, this will aid the interviewer in the selection of candidates.

1 Introduction

Interviews are becoming time-consuming affairs. Employees are required to travel to locations and conduct interviews, and it is difficult to manually remember each and every aspect of a candidate or the interview process. In many instances, the artificial intelligence system assists us in simplifying things.

Using the conventional method of recruitment, an organization's HR department invites individuals based on their resumes to an interview for a specific position. This HR department manually evaluates a candidate's skills based on their résumé to determine if he or she is qualified for the position or not. HR's conduct interviews, and the panel plays a significant role in determining who is the best applicant for the post. They examine not just the candidate's talents, but also his or her personality.

Maintaining resumes and profiles of all candidates becomes a very tedious job when it comes to big mass recruitment companies because they provide employment in bulk, and thus maintaining or storing data physically is not possible.

Machine Learning enables the path through which a computer can be trained to follow specific instructions again and again to make human life easy. The most common usage of machine learning is for the classification of objects. In machine learning, iteration is important because models are exposed to new data and adapt accordingly. Machine learning models learn from previous results and computation to produce correct and reliable decisions. In statistics, classification is a supervised learning concept in which segregation of data into similar groups is done depending upon various factors and observations.

A resume is one of the most important necessities when it comes to the selection of a candidate for any job. When the company's hiring team receives the resume of any candidate, the skills, if extracted using automation, will save a tremendous amount of time for the hiring team as they no longer need to sit and read through each and every word. But for this, first dataset needs to be scraped or made, then preprocessing has to be done on that dataset. Once the preprocessing of words is done, then using various machine learning algorithms, the data needs to be classified into different classes of job profile in accordance to the skillset...

2 Literature Survey

Finding the need for resume classification: As humans, everyone is bound to make mistakes. Storing and sharing physical copies safely is very inconvenient and inefficient. For this purpose of overcoming the drawbacks, need for resume classification significantly increases for which Artificial intelligence tools and Machine Learning algorithms [1] have been used widely.

Categorization of similar data together: With the advancement of computer and information technologies, a large number of research papers have been published both online and offline, and as new study topics continue to emerge, users are having a difficult time discovering and classifying their relevant research articles. A classification method that can cluster research articles into

* Corresponding author: palriya292@gmail.com

a meaningful class in which publications are extremely likely to have similar subjects is needed to overcome the restrictions. The suggested method uses K-means Clustering [2] and the Latent Dirichlet allocation (LDA) scheme [2] to extract representative keywords from the abstracts of each publication and subjects.

Importance of Automation: the issue regarding lack of automated systems for medical institutions and hospitals have caused a splurge for development of automation for hospital system [3]. Natural Language Processing (NLP)[11] benefits society as a whole as text-based information handling is very difficult manually.

Preprocessing of data using TF-IDF Vectorization: A well-developed categorization system that can group research papers into relevant classes based on their subjects [4] helps in finding out relevant data in the most time efficient manner. The suggested approach retrieves representative keywords from each paper's and topic's abstract. Then, using the Term frequency-inverse document frequency (TF-IDF) values [4] of each article, the K-means clustering method [4] is used to categorize the entire set of papers into research papers with comparable themes.

Modelling of semi-structured documents to fetch job postings from resume: Matching semi-structured resumes with positions in a big size real-world collection is a tough challenge. Experiments reveal that the SRM technique [5] yielded encouraging results and outperformed traditional unstructured relevance models in the first try. W. Bruce Croft, Xing Yi, and James Allan [5] Furthermore, we compared the suggested system's efficiency and efficacy to those of state-of-the-art online recruiting methods. A system that utilizes machine learning to match job postings and resumes for huge data sets; in our work, it is less difficult than previous papers; and using text mining, the extracted data from the resume is matched with the keyword recorded in the database and categorized for each job category.

3 Methodology

This section will describe the methodology and concepts that facilitate the building of classification model capable for resume classification and displaying the output with a suitable job profile for the candidate. The system works in the following phases as given below.

3.1 Data Gathering

Data Gathering includes collection of datasets from various websites like kaggle.com, glassdoor.com and indeed.com. The datasets are not classified and are unstructured datasets in which the data will be cleaned, classified, and stored in "25_cleaned_job_descriptions.csv" including some parts from Kaggle and some from glassdoor and indeed.com. 70% of the data is being used for training data and the remaining 30% will be used for test data.

3.2 Data Cleanup

The dataset containing a huge number of records is still very rough and unclassified. Data cleaning will be done by removing any blank spaces from the data, then changing all the text to lowercase to avoid confusion and removing stop words from the data. Stop words are those words which don't play an important role in sentence formation, such as "are", "we", "is", etc. Cleaned data is stored in a separate dataset containing 10,000 entries with two main classes of "query" and "description".

3.3 Tokenization

In this step, each entry in the corpus i.e., each entry in the document will be broken down into a set of words. To begin the tokenization process, we look for concepts or words that make up a character sequence. This is significant because we will be able to deduce meaning from the original text sequence using these terms. Tokenization is the process of separating large chunks of text into smaller pieces known as tokens. This is accomplished by deleting or isolating characters like as whitespace and punctuation. Tokens are phrases that are divided into individual words after being tokenized out of paragraphs. We may obtain information such as the number of words in a text, the frequency of a specific term in the text, and much more by doing Tokenization. Tokenization can be done in a variety of methods, such utilising the Natural Language Toolkit [NLTK], the spaCy library, and so on. Tokenization is a required step for subsequent text processing such as stop word removal, stemming, and lemmatization.

3.4 Stemming and Lemmatization

It is common to see a single English word employed in a variety of different ways in different phrases based on its grammatical rules. "Describe", "describing" and "described", for example, are all various tenses of the same verb. This condition necessitates the reduction of all changed or derived versions of a word to its primary stem or base, so that these derivationally related terms with comparable meanings are not deemed distinct from one another. Both stemming and lemmatization have the same goal but take different approaches to achieve it.

"Stemming is the mechanism of reducing inflected or derived words to their word root, or stem. It is a crude heuristic process that involves chopping off the ends of words to achieve this objective, and often includes the removal of derivational affixes [7]" These are rule-based algorithms that analyse a certain word under a variety of scenarios and then decide how to shorten it based on a list of recognized suffixes. It is worth noting that the root generated after stemming may not be the same as the word's morphological root. Stemming is prone to under and over-stemming. Porter-Stemmer, Snowball stemmer, and Lancaster stemmer are some

common stemming algorithms. Lemmatization, on the other hand, is the process of accurately reducing words to root words using a language dictionary. Lemmatization, as opposed to Stemming, which merely chops out tokens by basic pattern matching, is a more sophisticated technique that employs language vocabulary and morphological study of words to provide linguistically proper lemmas. This implies that lemmatization makes use of context information and may thus distinguish between words with various meanings based on parts of speech. For the English language, our system uses the NLTK python package's WordNet Lemmatizer (based on the WordNew Database).

The concept of stemming and lemmatization can be understood by a simple example. The word “loving” will turn into “lov” after stemming which has no meaning while if lemmatized, “loving” will turn into “love” which now has a proper meaning and use case.

3.5 Parts of Speech (POS) Tagging

It is the process of associating grammatical information with a word depending on its context and relationship to other words in the sentence [8]. According to its usage in the phrase, the part-of-speech tag identifies whether the word is a noun, pronoun, verb, adjective, or other. These tags must be assigned in order to grasp the right meaning of a phrase and to create knowledge bases for character recognition. This procedure is not as straightforward as mapping a word to its appropriate part of speech tags. This is because a word may have a distinct part of speech depending on the context in which it is spoken.

For example, "writing" is a Verb in the statement "I am writing an essay," yet "building" is a Noun in the line "I stay in the tallest building in the entire town." It is a supervised learning approach that analyses information such as the preceding word, following word, initial letter capitalised or not, and so on to label the words after tokenization. It is also known as grammatical tagging.

3.6 TF-IDF Vectorization

TF-IDF stands for Term Frequency – Inverse Document Frequency and it tells us how important a word is from the set of words in the dataset and assigns a tfidf value indicating the importance of the word as per frequency.

The formula for term frequency is the division of number of occurrences of a word in a sentence by the total number of words in the sentence (1). Inverse document frequency is calculated by the log of total sentences in the document divided by the sentences that actually contain the word (2). The multiplication of TF formula and IDF formula will give us a value in the form of a vector with a graph of importance on one hand and set of words on the other hand (3). Fig. 1 depicts a graph that shows TF-IDF vectorization which explains that the word with higher importance is placed according to ascending order of its importance and hence an inverse growth curve.

$$\text{TF} = \frac{No.of\ occurence\ in\ sentence}{Total\ no.of\ words\ in\ sentence} \quad (1)$$

$$\text{IDF} = \log\left[\frac{total\ sentences\ in\ document}{sentences\ which\ actually\ contain\ the\ word}\right] \quad (2)$$

$$\text{TF-IDF Vectorization} = \text{TF} \times \text{IDF} \quad (3)$$

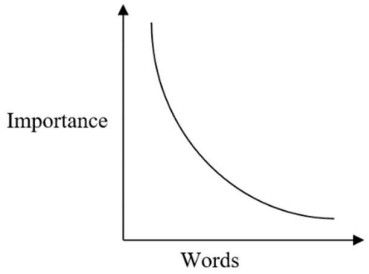

Fig. 1. Shows the graph of TF-IDF Vectorization

This will mark all the important sets of words related to jobs and skill sets with the highest term frequency value which can be then used to fetch important words and use them to train data on various algorithms.

3.7 Applying Classification Algorithm To Dataset

The following classification algorithms have been used for classification and model training.

Naïve bayes is a classification algorithm that works on probabilistic output as in whether the event is going to occur or not provided with a set of conditions. On the NB Classifier, the training data is fitted. Then, in the validation dataset labels are being predicted. To get accuracy, use accuracy_score function. This classification model gave an accuracy of around 45% and did not get the expected results.

Support Vector Machine (SVM) algorithm classifies data by drawing a hyperplane between two or more items. The Hyperplane which best classifies the items is considered as ideal output. The working flow of SVM is similar to that of Naïve Bayes. This classification model gave an accuracy of 60% which proved to be better than naïve bayes model.

Random Forest is a classification algorithm that works on the principle of decision trees. It takes in input of many decision trees and gives the best majority output from all the inputted decision trees. On the RF Classifier the training data is fitted. Then, in the validation dataset labels are being predicted. To get accuracy, use accuracy_score function. Random forest model gave an accuracy of 70% and gave the most correct predictions as per the comparison.

The comparison between the naïve bayes model, the SVM model and the random forest model with their accuracy, precision, recall and F1 score is given in TABLE 1.

4 Methodology Flowchart

The methodology is important to understand, and hence it becomes necessary to make a flowchart for easy understanding of the flow of system. Fig. 2. Shows the flowchart of the system.

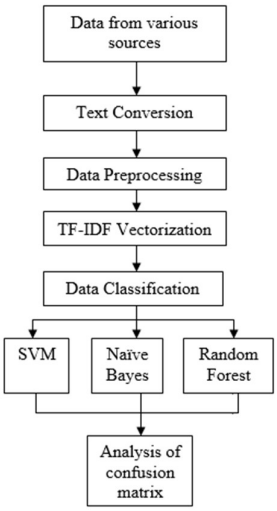

Fig. 2. Proposed Flowchart for the system

First the dataset needs to be scraped which can be done using various websites like indeed.com, glassdoor.com and kaggle.com. Once the dataset has been scrapped, preprocessing of data is to be done by doing proper stemming and lemmatization, removing stop words and filler words store the relevant words in a separate column for further process. After the preprocessing of data is done, making sure that the relevant data have the words which have occurred the highest amount of times need to be put in a term-frequency document using TF-IDF Vectorization. Once the previous steps are done, the cleaned data is now ready to test and form classification models using machine learning algorithms. Analysing the model by their accuracy and the forming confusion matrix for the same is required for better understanding of the results.

5 Results

The dataset has been taken from various websites like indeed.com, glassdoor.com and kaggle.com. These datasets are unstructured and thus preprocessing has been done which includes, tokenization, removing stop words, stemming and lemmatization with POS Tagging. The output for preprocessing data is shown in Fig. 3. Which consists of 3 main columns in its corpus having 10,000 rows. "Query" displaying all the job profiles (Fig. 4), "Description" containing raw data of words (Fig. 5), and "text_final" displaying the main set of preprocessed words (Fig. 6). The zoomed version of Fig 3. Has been represented in Fig. 4, Fig. 5, and Fig. 6.

Corpus

	Query	Description	text_final
0	Data Scientist	[job, description, junior, data, scientist, ib...	['job', 'description', 'junior', 'data', 'scie...
1	Data Scientist	[overall, summary, data, scientist, data, scie...	['overall', 'summary', 'data', 'scientist', 'd...
2	Data Scientist	[team, data, science, team, newly, formed, app...	['team', 'data', 'science', 'team', 'newly', '...
3	Data Scientist	[need, junior, data, scientist, ny, area, remo...	['need', 'junior', 'data', 'scientist', 'ny', ...
4	Data Scientist	[want, help, guide, core, business, spotify, u...	['want', 'help', 'guide', 'core', 'business', ...
...	...	...	...
9995	Network Architect	[opportunity, customer, understand, digital, t...	['opportunity', 'customer', 'understand', 'dig...
9996	Network Architect	[nasa, ames, research, center, ha, requirement...	['nasa', 'ames', 'research', 'center', 'ha', '...
9997	Network Architect	[ll, distinguished, engineer, proven, technol...	['distinguish', 'engineer', 'proven', 'technol...
9998	Network Architect	[software, development, engineer, full, stack,...	['software', 'development', 'engineer', 'full'...
9999	Network Architect	[architect, san, francisco, ca, leading, provi...	['architect', 'san', 'francisco', 'ca', 'lead'...

10000 rows × 3 columns

Fig. 3. Output of Preprocessing of Data

Corpus

	Query
0	Data Scientist
1	Data Scientist
2	Data Scientist
3	Data Scientist
4	Data Scientist
...	...
9995	Network Architect
9996	Network Architect
9997	Network Architect
9998	Network Architect
9999	Network Architect

10000 rows × 3 columns

Fig. 4. Query Column from Fig. 3.

Description
[job, description, junior, data, scientist, ib...
[overall, summary, data, scientist, data, scie...
[team, data, science, team, newly, formed, app...
[need, junior, data, scientist, ny, area, remo...
[want, help, guide, core, business, spotify, u...
...
[opportunity, customer, understand, digital, t...
[nasa, ames, research, center, ha, requirement...
[ll, distinguished, engineer, proven, technol...
[software, development, engineer, full, stack,...
[architect, san, francisco, ca, leading, provi...

Fig. 5. Description Column from Fig. 3.

text_final
['job', 'description', 'junior', 'data', 'scie...
['overall', 'summary', 'data', 'scientist', 'd...
['team', 'data', 'science', 'team', 'newly', '...
['need', 'junior', 'data', 'scientist', 'ny', ...
['want', 'help', 'guide', 'core', 'business', ...
...
['opportunity', 'customer', 'understand', 'dig...
['nasa', 'ames', 'research', 'center', 'ha', '...
['distinguish', 'engineer', 'proven', 'technol...
['software', 'development', 'engineer', 'full'...
['architect', 'san', 'francisco', 'ca', 'lead'...

Fig. 6. text_final Column from Fig. 3.

After preprocessing of data, the dataset containing all the words are given TF-IDF values in order to arrange the

word matrix in ascending order of their importance. Result of TF-IDF vectorization shows columns with their frequency number which have term frequency of more than 5000 is shown in Fig. 7. With output like: 'job': 2456, 'description': 1221, 'junior': 2480, 'data': 1134, 'scientist': 3984, 'ibm': 2175, 'work': 4959, 'part': 3216, 'team': 4448, 'solve': 4182, 'problem': 3498, 'use': 4763, 'technique': 4455, 'apply': 266, 'scientific': 3983, 'method': 2816, 'business': 578, 'scenario': 3971, 'clean': 734, 'prepare': 3445, 'statistical': 4286, 'machine': 2667, 'learn': 2556, 'model': 2877, 'variety': 4798, 'analytical': 215, 'algorithm': 165, 'build': 570,…

```
print(Tfidf_vect.vocabulary_)
```

```
{'job': 2456, 'description': 1221, 'junior': 2480, 'data': 1134, 'scientist': 3984,
```

```
print(Tfidf_vect.vocabulary_)
```

```
'scientist': 3984, 'ibm': 2175, 'work': 4959, 'part': 3216, 'team': 4448, 'solve': 4182,
```

Fig. 7. Output of TD-IDF Vectorization

Data after being cleaned into a new dataset containing 10,000 entries with two main categories “query” and “description”. Fig. 8. Depicts the main 25 job classes that are being evaluated which are: 'Artificial Intelligence', 'Big Data Engineer', 'Business Analyst', 'Business Intelligence Analyst', 'Cloud Architect', 'Cloud Services Developer', 'Data Analyst', 'Data Architect', 'Data Engineer', 'Data Quality Manager', 'Data Scientist', 'Data Visualization Expert', 'Data Warehousing', 'Data and Analytics Manager', 'Database Administrator', 'Deep Learning', 'Full Stack Developer', 'IT Consultant', 'IT Systems Administrator', 'Information Security Analyst', 'Machine Learning', 'Network Architect', 'Statistics', 'Technical Operations', 'Technology Integration'.

```
array(['Artificial Intelligence', 'Big Data Engineer', 'Business Analyst',
       'Business Intelligence Analyst', 'Cloud Architect',
       'Cloud Services Developer', 'Data Analyst', 'Data Architect',
       'Data Engineer', 'Data Quality Manager', 'Data Scientist',
       'Data Visualization Expert', 'Data Warehousing',
       'Data and Analytics Manager', 'Database Administrator',
       'Deep Learning', 'Full Stack Developer', 'IT Consultant',
       'IT Systems Administrator', 'Information Security Analyst',
       'Machine Learning', 'Network Architect', 'Statistics',
       'Technical Operations', 'Technology Integration'], dtype=object)
```

Fig. 8. Classification of jobs

After the data is cleaned and arranged, classification of data into various job profiles and skill sets can be achieved using various machine learning classification algorithms. The developed systems have excellent true predictions when confusion matrix is taken into consideration. The following diagrams below depict the confusion matrix of various classifications, where true positive values are those values where the model predicted the correct job profiles as expected value. Fig. 9 shows the confusion matrix of all the job classes.

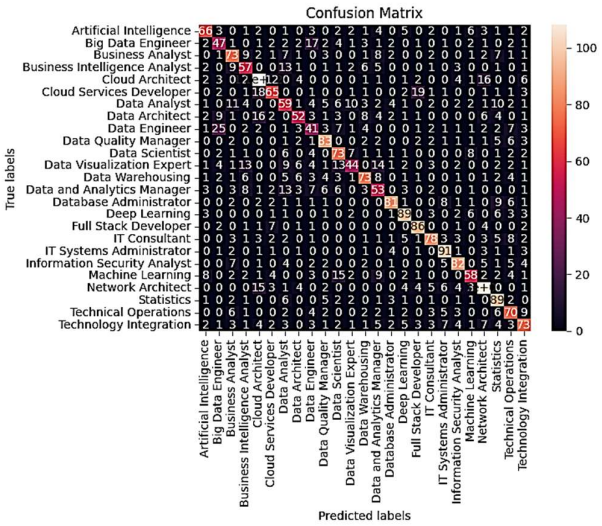

Fig. 9. Confusion matrix of job classes

The confusion matrix for naïve bayes classification model is shown in Fig. 10 depicting the true positive, true negative, false positive and false negative values that have been detected. A diagonal line of dark colored boxes can be observed which shows the true positive values for each label using Naïve Bayes Classification.

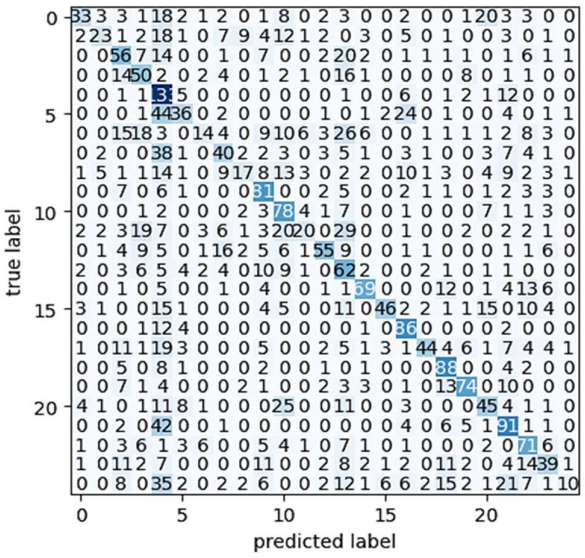

Fig. 10. Confusion matrix for naïve bayes classification

The confusion matrix for SVM classification model is shown in Fig. 11 depicting the true positive, true negative, false positive and false negative values that have been detected. A diagonal line of dark colored boxes can be observed which shows the true positive values for each label using SVM classification.

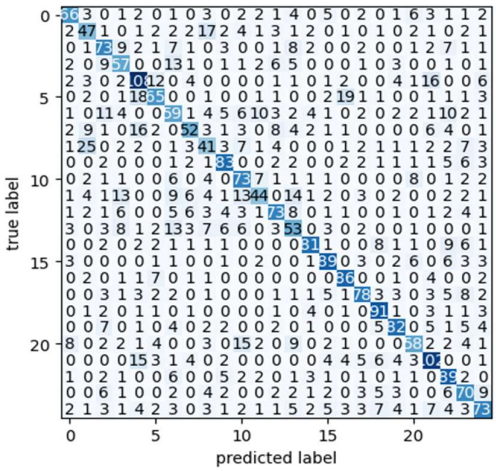

Fig. 11. Confusion matrix for SVM Classification

The confusion matrix for Random Forest is shown in Fig.12 depicting the true positive, true negative, false positive and false negative values have detected. A diagonal line of dark colored boxes can be observed which shows the true positive values for each label using random forest classification.

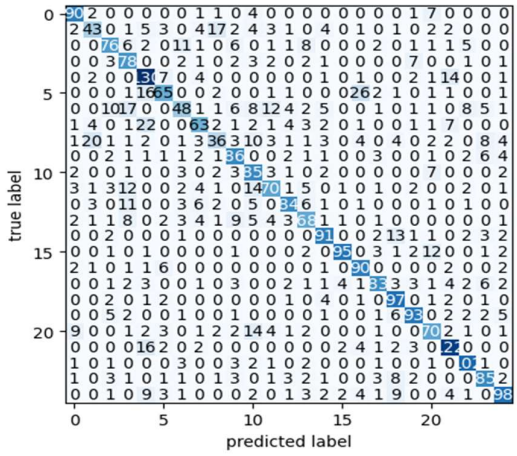

Fig. 12. Confusion matrix for random forest classification

The end result of the confusion matrix and classification models shows that Random Forest is best suited for classification of resume data with 2500 decision trees taken into consideration. TABLE 1. shows the result and comparison of various classification algorithms.

Table 1. Comparison of various classification algorithms

Algorithm	Accuracy	Precision	Recall	F1 Score
Naïve Bayes	45	0.521	0.452	0.448
SVM	60	0.598	0.597	0.594
Random Forest	70	0.687	0.683	0.678

Despite of average accuracy being 70%, the confusion matrix shows excellent results. Also, accuracy is not the only measure that needs to be taken into consideration while building a model but the confusion matrix plays an equally important part for accurate results for how many true positive values have been returned. As per the comparison in TABLE 1, Random forest has the highest accuracy and has the best confusion matrix.

The random forest accuracy can be improved by increasing the number of trees or by increasing the size of data. In our case, we took 2,500 trees and increasing the number would increase the accuracy of the model but at the same time the learning time would get slower. Another technique that can be applied for improvement is to use hyper parameter tuning for random forest classifier.

Conclusion and Future Work

The concept of classification is grasped by Resume Classification, and classification models have been built using numerous techniques. This resume categorization platform will make the e-recruitment process more efficient and user-friendly. This approach will assist businesses and save time throughout the recruitment process.

Future work includes combining these classification models with frontend resume portal where the user will be able to upload their resume and resume classification can work in the backend. From the results and discussions, we can conclude that various algorithms have given different accuracy score percentages with Random Forest having the highest accuracy among all three of them. Also, the confusion matrix of random forest is well suited for all the observed classes.

Thereby, resume classification is achieved using preprocessing of data, cleaning of data and by applying various classification algorithms.

This research paper can also be extended with the combination of other video interviewing features such as facial recognition, voice to text generation and voice analysis. Building a plugin for people to include resume classification models into their project is also one of the future goals of this research paper and project work.

References

1. B. Balci, D. Saadati, D. Shiferaw, Handwritten Text Recognition using Deep Learning, STAN. U. (2017).
2. SW. Kim, JM. Gil, Research paper classification systems based on TF-IDF and LDA schemes. Hum. Cent. Comput. Inf. Sci. **9**, 30 (2019).
3. C. Friedman, G. Hripcsak, Natural language processing and its future in medicine. Acad Med. August (1999)
4. J. Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, Rutgers University.
5. E. Xing Yi, James Allan and W. Bruce Croft Center for Intelligent Information Retrieval, Department of

Computer Science 140 Governor"s Drive, Matching Resumes and Jobs Based on Relevance Models, SIGIR, (2017)

6. F. Aseel B. Kmail, Mohammed Maree, Mohammed Belkhatir, Saadat M. Alhashmi -An Automatic Online Recruitment System based on Exploiting Multiple Semantic Resources and Concept-relatedness Measures, IEEE 27th International Conference on Tools with artificial intelligence (2015)
7. Jivani, A.G., A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl, (2011)
8. Gelbukh, A., Computational Linguistics And Intelligent Text Processing. Berlin, Heidelberg: Springer Berlin Heidelberg (2014)
9. Laumer, S. and Eckhardt, A., May. Help to find the needle in a haystack: integrating recommender systems in an IT supported staff recruitment system. In Proceedings of the special interest group on management information system's 47th annual conference on Computer personnel research, (2009).
10. A. Zaroor, M. Maree, MuathSabhaJRC, A job post and resume classification system for online recruitment, International Conference on tools with Artifical Intelligence, (2017)
11. Bird, S., Klein, E. and Loper, E, Natural Language Processing With Python. Bejing: O'Reilly, (2009).
12. Kim S. B., Rim H. C., Yook D. S. and Lim H. S., "Effective Methods for Improving Naive Bayes Text Classifiers", (2002)
13. F. Dernoncourt and J. Y. Lee, Pubmed 200k rct, a dataset for sequential sentence classification in medical abstracts (2017).
14. D. E. Johnson, F. J. Oles, T. Zhang, T. Goetz, A decision-tree-based symbolic rule induction system for text categorization, IBM Systems Journal, September (2002).
15. S. Sanyal, N. Ghosh, S. Hazra, S. Adhikary, Resume Parser with Natural language Processing, IJESC (2007).
16. X.Chen,X. Lin, Big Data Deep Learning: Challenges and Perspectives, IEEE, (2014)
17. Bao Y. and Ishii N., Combining Multiple kNN Classifiers for Text Categorization by Reducts.
18. Kolluri, Johnson and Razia, Shaik and Nayak, Soumya Ranjan, Text Classification Using Machine Learning and Deep Learning Models. International Conference on Artificial Intelligence in Manufacturing & Renewable Energy (ICAIMRE) (2019)
19. S. Brindha, K. Prabha and S. Sukumaran, A survey on classification techniques for text mining, 3rd International Conference on Advanced Computing and Communication Systems (ICACCS) (2016)