2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT) 10-11, December 2021

Screening and Ranking Resumes using Stacked Model

Rasika Ransing
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
rasikaransing275@gmail.com

Akshaya Mohan
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
akshaya.mohan@vit.edu.in

Nikita Bhrugumaharshi Emberi
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
nikita.emberi@gmail.com

Kailas Mahavarkar
Department of Information Technology
Vidyalankar Institute of Technology
Mumbai, India
kailas.mahavarkar@vit.edu.in

***Abstract*—Talent acquisition is essential for all companies irrespective of the size of their business. As it is next to impossible to look through numerous resumes manually, we have created an automated resume screening application. This system makes use of Machine Learning algorithms such as KNN, Linear SVC, and XGBoost. A two-level stacked model containing all these algorithms is constructed which helps in predicting specific job profiles from a text description accurately. This framework can be valuable for organizations to waitlist competitors and furthermore for the applicants who can check if their resume is very much shaped for the system to recognize right work profiles from it. A ranking system is also implemented, for the companies, featuring the most relevant profiles on the top.**

Keywords— Stacking Classifiers, Linear SVC, XGBoost Classifier,SVC,KNN, Machine Learning, Resume Screening, Resume Ranking, Resume Evaluation, Natural Language Processing.

I. INTRODUCTION

Recruitment is an ancient process, from paper applications to online resumes this process has undergone various transitions. With a rapid increase in the population, Any enterprise having a job opening for a specific position will have their mailboxes flooded with tens of thousands of emails from aspiring candidates each day.

Screening through this overwhelming information to find skilled and qualified candidates is a time-consuming and frustrating process, irrespective of the size of the business. To perform effective screening, one must have a cognitive understanding of different job roles to filter out candidates that are a right fit. With an immense variety of different job roles, these days and a massive number of applications received, it only worsens the process of manual screening. Being able to filter irrelevant profiles and rank them according to the requirement will save time as well as money.

The objective of drafting this paper is to address the above issues using different machine learning algorithms. This approach tends to the recruiters' needs by filtering out resumes based on whatever requirement is mentioned by them as well as an individual can verify the relevance of their resume for a particular IT role. After filtering out relevant resumes a ranking system is provided which recommends a better resume for a given textual job description. The impact of these features and validity of different classification models is examined by using a two-level stacking scheme of base models which provide favourable results in comparison with individual models.

II LITERATURE SURVEY

Peng Xu and Denilson Barbosa [1] evaluated the method of comparing resumes to job specifications considering severalmodels like random forest, gradient boosting tree and densely connected neural networks individually and stacked and suggested that using stacked models show better accuracy and consistently outmatches standard methods, including neural networks. However, the implementation does not look for difficult cases. e.g., where the job requires experience ondifferent technologies (for eg, data analytics) and the resume mentions command over a specific tool (like, R).

Sushruta Mishra et. Al [2] proposed an approach that consists of two functional components. Stacked KNN Learner Modulemakes use of five variants of KNN which include (1, 3, 5, 7, 9)-NN. The candidate's data's individual prediction from these different variants of KNN algorithm is put in to the Hard Voting Ensemble Predictive units which gives an aggregate of most predicted class votes found for each class label to decide upon the candidate's relevance for an applied IT-based job.

KNN algorithm was implemented by Riza Tanaz Fareed et al. [3] to cluster the resumes with respect to categories and Cosine Similarity to analyze the closeness or similarity between candidate's resume and the original job description, and ranking them accordingly. The model accepts resumes in CSV format only, where as most of the resumes are in .pdf, .doc, and different formats. The summarization by compressing the text may result in the loss of valuable information while using the "gensim" package for creating the summary in python.

Momin Adnan et al. [4] suggested a Linear SVM classifier-based approach for screening candidates for recruitment. The model uses cosine similarity, KNN, and content-based Recommendation to find resumes relevant to the job description. While most resumes are in.pdf,.doc and other forms, the model only takes resumes in CSV format. The implicit compression of the text due to summarising may have resulted in the loss of valuable information while using the "gensim" package to construct a summary.

Frank Färber et. al. [5] proposed to Personnel Recruitment Selection, with his method of Automated Recommendation Approach(ARA) which started by providing a blueprint of a matching method based on a probabilistic ARA, which then went on to give some favourable results using this algorithm. The ARA's full potential was assumed to yet be realised because of little training data.

V.V. Dixit et. al. [6] proposed a project which will sort all the resumes according to the requirement of the company and forward it to the respective HR for further recruitment process using Artificial Intelligence technology. The required resume is filtered among the huge numbers of candidates.

Chirag Daryani et. al [7] proposed a system that employed Natural Language Processing to extract useful information like education, experience, skills, etc. from the resumes which hasn't been structured and hence creates a summarised form of each application. The proposed solution makes use of cosine similarity and follows a vectorisation model and to compare each resume with the job description. The ranking scores after proper calculations can then be utilised to determine appropriate candidates for that particular job opening. However, "cold start" problem, appearing due to data scantiness, is particularly difficult here: emerging skills and technologies arrive often which frequently lead to sparse vectors [1]

EXPERT was presented by Senthil Kumaran and A. Sankar [8], which is a smart tool for filtering the candidates for hiring using structured mapping and it has three phases. In the starting phase, the system accumulates all resumes of candidates and creates structured documents. Job specification or openings are represented as ontology in the next phases, Expert maps the process requirement ontology onto the candidate ontology report and retrieves the eligible applicants.

III. METHODOLOGY

A. *Dataset description:*

Our dataset was taken from Kaggle. The data is in CSV format, with two columns which were Role and Input.

Role: the category to which the resume belongs

Input: the complete resume of the candidate.

The dataset used was confined to the IT sector. It had 10 job descriptions (Role) namely and Project Manager, Database Administrator, Security Analyst, Java Developer, Python Developer, Network Administrator, Systems Administrator, Data Scientist, Web Developer, Software Developer and 2000 resumes (Input).

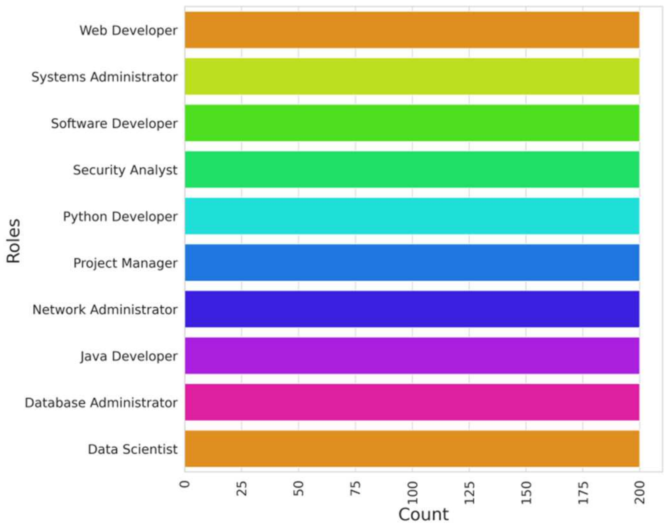


Fig. 1 Data distributed over the distinct roles

B. *Pre-processing*

In this procedure the resumes obtained as a input will undergo some cleansing to eliminate special characters or junk characters that might be present. All special characters, punctuations, new lines, blank spaces, special mentions, hash tags and so on are eliminated during this process.

Eliminating Stop Words and Performing Tokenization:

The Natural Language Toolkit (NLTK) is a library used to help the system learn, analyse, and associate what words describe a particular job description the most.

Stop Words are the words that occur most frequently in the English sentence that do not add any value to the meaning of the sentence and are only used to add grammatical value. For Example, words such as "the," "is" and "and" would easily qualify as stop words. Stop words are eliminated using NLTK that contain a total of 179 stop words.

Tokenization is a process of splitting large text into words. This is a requirement in Natural Language Processing where each word is captured and subjected to additional analysis like classifying and counting them.

Following was the procedure employed to tokenize and remove stop words from the resumes of the candidates:

1. All the stop words from nltk corpus were stored in a list called 'stop_words'.
2. Nltk.word_tokenize() is the command used to split the words from each of the resumes provided
3. Cleaned Text obtained from procedure 4.1 was tokenized and stored in an array.
4. Tokenized words appearing in the list of stop_words were extracted and resultant words were stored in a final list of "total_words".

The top 50 most frequently used words chosen described each job description efficiently. These words helped train our model and provide satisfactory results.

We plotted Word Cloud to visualize the most common occurring words presented in fig. 2

Web
development
design
application
Developer
End
year
using
JavaScript
Experience
website
Front
Developed
management
software
jQuery
CSS
system
HTML
Bootstrap
Work
service
user
project
client
UI
Designer
new
API
code
database
server
React
HTML5
Worked
marketing
Created
Skill
Used
Designed
page
JS
PHP
site
data
team
Angular
WordPress
content
CSS3

Fig. 2 Word Cloud Representing Most Common occurring words

Encoding Roles using Label Encoding:

Label encoding is a process of converting labels into numeric format (which a machine can understand). Even though the "Role" column is 'Nominal' data, we are using Label Encoding because the 'Role' column is our 'target' column. Each category became a class as a result of Label Encoding, and we were able to create a multiclass classification model.. Another column named "Correct Roles" was created after cleaning the resume.

Label encoding provides a number for all the Roles as presented in Fig. 3

Labels	Encoded Labels
Data Scientist	0
Database Admin	1
Java Developer	2
Network Admin	3
Project Manager	4
Python Developer	5
Security Analyst	6
Software Developer	7
Systems Admin	8
Web Developer	9

Fig. 3 Encoded labels for each role

C. Model Implementation

We used the TF-IDF [9] method to convert resume text into the vector form to normalize the frequency of tokens in a resume document with respect to the rest of the corpus.

Initially, the classification was done using Linear SVC, KNN and XGBoost separately. Both were able to provide the probability of the predicted class as expected, which eventually helped for further stacking.

Following is the procedure implemented to perform KNN, Linear SVC and XGBoost algorithms:

We used the technique of pipelining for all the above models. Our pipelining consists of 2 stages:

- First Stage: TF-IDF Transformer for vectorizing the text
- Second Stage: Estimator which Involves either individual classifiers or stacked models.

k-nearest neighbors (KNN) [10]: KNN is a supervised learning algorithm that makes use of input value (labelled) set to predict the output of data points. This algorithm makes highly accurate predictions. The parameter passed for the k-neighbors classifier include algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, all of which are default values except n_neighbors which was set to 13. All of this was stored in a variable called "text_clf"

Linear SVC [11]: Linear SVC makes use of a linear kernel function to perform nicely with huge amount of samples and perform classification. It has additional parameters like penalty normalisation. The parameters passed for this algorithm include kernel='linear' to perform Linear SVC, probability=True to enable distribution of probability among roles applicable to a particular job description, random_state = 0. All of this was stored in a variable called "text_clf_lsvc".

XGBoost Classifier[12]: stands for eXtreme Gradient Boosting. It is a specific implementation of Gradient Boosting whose sole purpose is to increase computational speed and improve the Model's performance. The parameters passed for this Classifier include "objective":"binary:logistic", 'alpha': 10, 'learning_rate': 0.1, all of which are Default values except 'colsample_bytree': 0.3, and 'max_depth' which is set to 5 to maximise Tree depth for base learners.

Stacking Classifiers [13]: Each of the above individual classifiers where then stacked to obtain optimum results required for ranking the resumes.

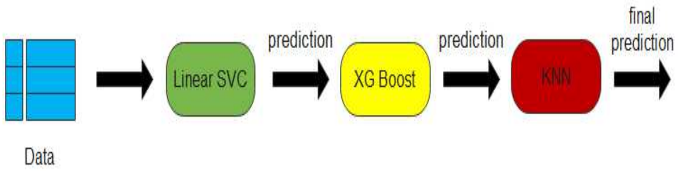


Fig 4. stacking classifier architecture implemented

Our stacking model's architecture presented in Fig.4 consists of base models, also known as Level-0 models, and a meta-model that combines the predictions of the base models, also known as a Level 1 model.

- Level 0 Model (Base-Model): Training data is fitted to these models and predictions from these models are compiled.
- Level 1 Model (Meta-Model): A model that employs the most efficient technique of combining the predictions of the meta models.

.

We initially implemented Linear SVC as our base learner (Level-0) to remove unwanted classes. We created a Stacking Classifier using Linear SVC as base estimator and a second layer as the final estimator. This Second layer (Level-1) consists of XGBoost Classifier as the base estimator and KNN as the final estimator to create the full pipeline.

D. Resume Filtering

After training the models the next step to be focussed on was filtering and ranking the resumes. We took the text specifying job requirements related to one of the roles present in our dataset (Presented in Fig. 5) from online job portals like Indeed. We predicted the roles applicable to the specified requirement using probability distribution [14] and stored the probabilities predicted with respect to each role in a dictionary with keys as encoded labels of roles and values as probabilities as:

predicted_probability = $\{R_i : P_i\}$ (1)
where,
R_i : Encoded labels of roles present in our dataset
P_i : Probability allocated by our model

From this dictionary (predicted_probability), predicted labels(ri) having probability greater than 0 were stored in the dictionary named "required_roles" in the format specified in (1). We appended a column called "probability distribution" to our dataset with the format specified in (1) and filtered the resumes having probabilities allocated to the labels predicted from the text of job requirement.

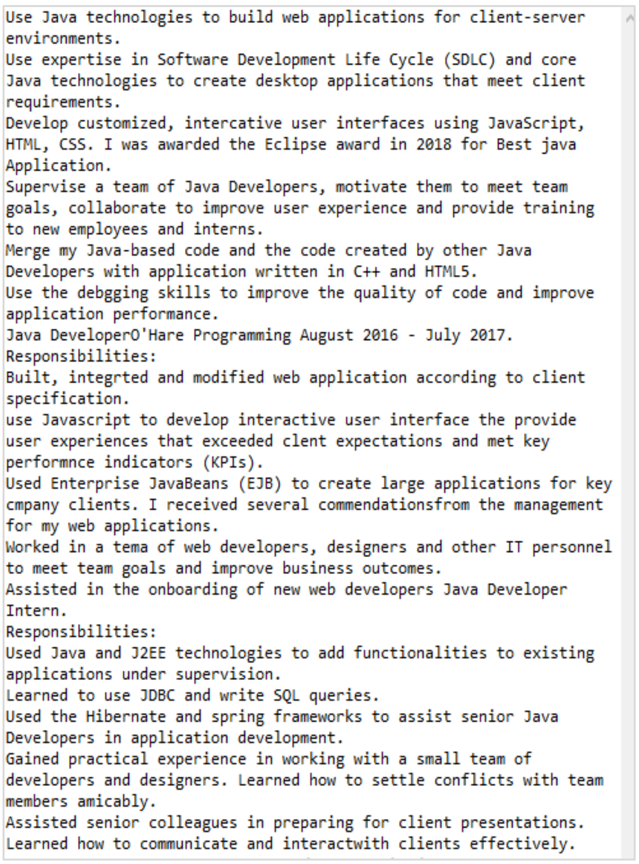
E. Resume Ranking

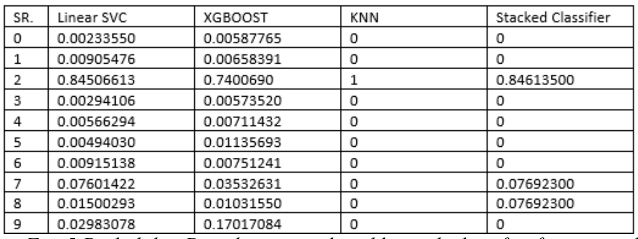
As discussed in Resume Filtering, to check if the job requirement text is related to the roles present in our dataset, we used the formula specified below:

$$factor1 = \sum_{0}^{length\ of\ context} (x)^2$$

$$factor2 = \left(\frac{1}{len(context)}\right)$$

$$relative\ score = (factor1 \times 2) + \ factor2$$


```
Use Java technologies to build web applications for client-server
environments.
Use expertise in Software Development Life Cycle (SDLC) and core
Java technologies to create desktop applications that meet client
requirements.
Develop customized, intercative user interfaces using JavaScript,
HTML, CSS. I was awarded the Eclipse award in 2018 for Best java
Application.
Supervise a team of Java Developers, motivate them to meet team
goals, collaborate to improve user experience and provide training
to new employees and interns.
Merge my Java-based code and the code created by other Java
Developers with application written in C++ and HTML5.
Use the debgging skills to improve the quality of code and improve
application performance.
Java DeveloperO'Hare Programming August 2016 - July 2017.
Responsibilities:
Built, integrted and modified web application according to client
specification.
use Javascript to develop interactive user interface the provide
user experiences that exceeded clent expectations and met key
performnce indicators (KPIs).
Used Enterprise JavaBeans (EJB) to create large applications for key
cmpany clients. I received several commendationsfrom the management
for my web applications.
Worked in a tema of web developers, designers and other IT personnel
to meet team goals and improve business outcomes.
Assisted in the onboarding of new web developers Java Developer
Intern.
Responsibilities:
Used Java and J2EE technologies to add functionalities to existing
applications under supervision.
Learned to use JDBC and write SQL queries.
Used the Hibernate and spring frameworks to assist senior Java
Developers in application development.
Gained practical experience in working with a small team of
developers and designers. Learned how to settle conflicts with team
members amicably.
Assisted senior colleagues in preparing for client presentations.
Learned how to communicate and interactwith clients effectively.
```



SR.	Linear SVC	XGBOOST	KNN	Stacked Classifier
0	0.00233550	0.00587765	0	0
1	0.00905476	0.00658391	0	0
2	0.84506613	0.7400690	1	0.84613500
3	0.00294106	0.00573520	0	0
4	0.00566294	0.00711432	0	0
5	0.00494030	0.01135693	0	0
6	0.00915138	0.00751241	0	0
7	0.07601422	0.03532631	0	0.07692300
8	0.01500293	0.01031550	0	0.07692300
9	0.02983078	0.17017084	0	0

Fig. 5 Probability Distribution predicted by each classifier for given job Description

model result for single test case = [0, 0, 0.07692308, 0, 0, 0, 0, 0.23076923, 0, 0.69230769]

after removing null values we get:
context =[0.07692308, 0.23076923, 0.69230769]

length of context is 3

factor1:
- sum of squares of each context value
- sum of [0.00591715976331361, 0.053254437869822494, 0.4792899408284023]
- factor1 score is 0.5384615384615384

factor2
- inverse of length of context i.e $\frac{1}{\text{length of context}}$
- factor2 is $\frac{1}{3}$ where 3 is length of context
- factor1 score is 0.3333333333333333

relative score = (factor1 x2) + factor 2

final relative score = 1.6794871794871793

to calculate final relative score we multiplied factor1 by 2(since we observed it added more value to test cases) then we added factor1 score with factor2 score.

If the relative frequency was found to be greater than a threshold of "1", then the further process ranking resumes were carried out. If the probability distribution obtained from the requirements mentioned by a company matches more than one job profile the resumes will be ranked in the following manner:

- Probability distribution calculated from each resume is stored in the column called "Probability distribution". This column was matched with the keys of the dictionary "Required_roles" and the matching resumes probabilities (which adhered to the threshold) were added and filtered.
- These filtered Resumes were sorted in descending order of added probabilities and a rank was assigned to each resume

IV. RESULTS AND ANALYSIS

We evaluated our model using stratified 10-fold cross validation, accuracy acheived from various classifiers and sklearn metrics accuracy score as presented in Table 1. Through a stratified 10 cross validation method, we noticed that our proposed ensemble model performed much better with a 90-10 training and testing ratio in comparison to others. These training and testing sets created are passed through the pipelines created for individual classifiers and stacked models for further evaluation. Comparison between the accuracy systems is shown in the equation.

TABLE I. PERFORMANCE EVALUATION

Model	Accuracy Score	Cross-validated average score
Linear SVC	0.835	0.7675
KNN	0.72	0.6565
XGBoost	0.85	0.7523
Stacked Classifier	0.83	0.73

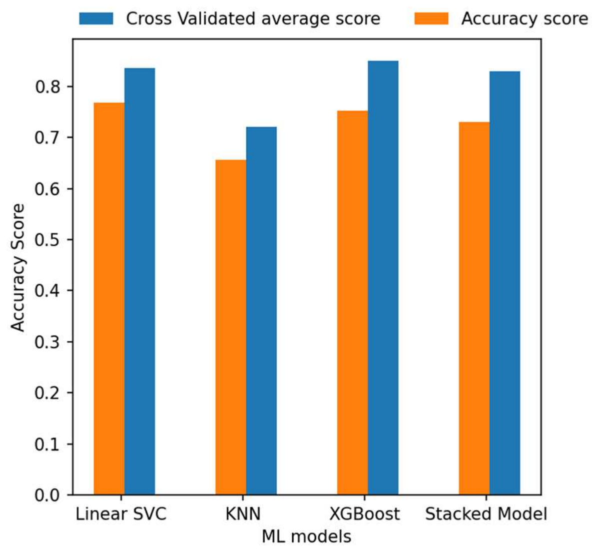


Fig. 6. Comparison between stratified 10-fold cross-validation average score and metrics accuracy scores for different classifiers

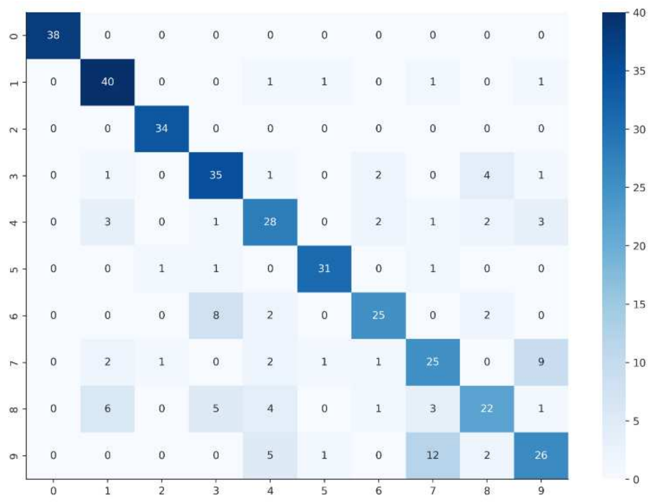


Fig. 7. Confusion matrix using Stacking Classifier

On initial analysis, Linear SVC and XGBoost gave us the maximum accuracy and a favourable confusion matrix. Both were able to predict the correct Role for a particular job description given to the system. The problem occurred while finding the probability distribution of the predicted results. These two algorithms provided a probability to all the roles irrespective of how small the number was which created a problem during ranking our resumes using the method discussed in Resume Filtering and Resume Screening. KNN algorithm gave us a poor accuracy, but the probability distribution was apt to suit our needs.

To not settle with a low accuracy from KNN and to find a suitable algorithm which provides an optimum probability distribution that matches our needs, we implemented Stacking Classifier. Probability distribution method helped us in predicting the possibility of a job requirement text being suitable for more than one job profile. As shown in the fig 5, the job description suits the profile of Java Developer, Software Developer and Systems Admin. Our stacked model with accuracy combined with SVC and XGBoost could predict both these roles correctly with the KNN as our final estimator as compared to SVC and XGBoost which assigned probability to every role irrespective of how small the probability is. These models individually did not meet our needs although the accuracy of SVC and XGBoost was slightly greater than that of Stacked Model.

We stacked all the above individual models as shown in Fig 4 to obtain an accuracy of 83% with roles getting a probability of 0 if the number is insignificant. Continuing with our highest performing model (Stacking Classifier), taking a look at the confusion matrix as shown in Fig 7 and show the variation between predicted and actual labels.

V. CONCLUSION AND FUTURE SCOPE

The proposed system has been successfully implemented using supervised learning and is yielding an accuracy of 83% utilizing stacking classifiers. The initial analysis performed using individual algorithms (Linear SVC, XG Boost, KNN) did not yield acceptable outcomes so we combined these algorithms using stacking. This system will assist the employers to screen candidates based on their specification and rank them in an orderly fashion.

This will make the hiring process simpler and ease the burden on employers as it is no longer required to screen through resumes of a large pool of candidates.

The proposed system can be deployed by integrating it with web application onhigh-performance computing platforms which are cloud based that provides a section for recruiters to specify requirements and filter the resumes stored in our database based on their rank and another section where individuals can upload resume to find the relevance of their resume to a particular job title.

The model could be upgraded to retrieve various helpful features from the text description in the resumes and the job specification using the techniques like Information Retrieval (IR) and Natural Language Processing whereby job postings and resumes are broken down into different categories (certifications, skills, experience, etc.), and uses IR-based ranking for candidates for jobs

The performance of the model could be enhanced further by using Deep Learning Models like Long-Short Term Memory, Recurrent Neural Network, or Convolutional Neural Network and others to enable semantic translation of phrases found in resumes and job descriptions. In a comparable manner, it would be exciting to train a model with neural language on the candidate's resumes who were recruited with the intention of soft talents that are desired by the company that aren't often articulated in the job description.
Another area of future work will be considering not only roles related to IT but also roles covering various domains like health and science, Banking and Finances, Management roles, etc. We were unable to reach that target due to the size of our dataset.

VI. ACKNOWLEDGMENTS

This project is been developed under the guidance of Prof. Rasika Ransing, Assistant Professor, Vidyalankar Institute of Technology, Mumbai.
Also, acknowledgement to all those who helped by providing
the valuable information in their respective field.

REFERENCES

[1] Xu P., Barbosa D. (2018) Matching Résumés to Job Descriptions with Stacked Models. In: Bagheri E., Cheung J. (eds) Advances in Artificial Intelligence. Canadian AI 2018. Lecture Notes in Computer Science, vol 10832. Springer, Cham. https://doi.org/10.1007/978-3-319-89656-4_31

[2] Mishra, Sushruta & Mallick, Pradeep Kumar & Tripathy, Hrudaya & Jena, Lambodar & Chae, Gyoo-Soo. (2021). Stacked KNN with hard voting predictive approach to assist hiring process in IT organizations. The International Journal of Electrical Engineering & Education. 002072092198901. 10.1177/0020720921989015.

[3] Rajath V , Riza Tanaz Fareed , Sharadadevi Kaganurmath, 2021, Resume Classification and Ranking using KNN and Cosine Similarity, international journal of engineering research & technology (IJERT) Volume 10, Issue 08 (August 2021)

[4] Momin Adnan, Gunduka Rakesh, Juneja Afza, Rakesh Narsayya Godavari, Gunduka and Zainul Abideen Mohd Sadiq Naseem., Resume Ranking using NLP and Machine Learning, (2016b). Institutional Repository of the Anjuman-I-Islams Kalsekar Technical Campus. https://core.ac.uk/display/55305289

[5] Färber, Frank & Weitzel, Tim & Keim, Tobias. (2003). An Automated Recommendation Approach to Selection in Personnel Recruitment.. 302.

[6] V. V. Dixit, Trisha Patel, Nidhi Deshpande, Kamini Sonawane. (2019). Resume Sorting using Artificial Intelligence

[7] Daryani, Chirag & Chhabra, Gurneet & Patel, Harsh & Chhabra, Indrajeet & Patel, Ruchi. (2020). An automated resume screening system using natural language processing and similarity. 99-103. 10.26480/etit.02.2020.99.103.

[8] V, Senthil kumaran & Annamalai, Sankar. (2013). Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping EXPERT. International Journal of Metadata, Semantics and Ontologies. 8. 56-64. 10.1504/IJMSO.2013.054184.

[9] Qaiser, Shahzad & Ali, Ramsha. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications. 181. 10.5120/ijca2018917395.

[10] Wang, Lishan. (2019). Research and Implementation of Machine Learning Classifier Based on KNN. IOP Conference Series: Materials Science and Engineering. 677. 052038. 10.1088/1757-899X/677/5/052038.

[11] K. Z. Arreola, J. Fehr and H. Burkhardt, "Fast Support Vector Machine Classification using linear SVMs," 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 366-369, doi: 10.1109/ICPR.2006.549.

[12] Santhanam, Ramraj & Uzir, Nishant & Raman, Sunil & Banerjee, Shatadeep. (2017). Experimenting XGBoost Algorithm for Prediction and Classification of Different Datasets.

[13] Ashish Patel.https://medium.com/ml-research-lab/stacking-ensemble-meta-algorithms-for-improve-predictions-f4b4cf3b9237

[14] Diego Lopez Yse. (2020). https://towardsdatascience.com/before-probability-distributions-d8a2f36b1cb