Automating Resume Screening and Job Description Matching

Ashirvad Das*, Ketan Kansal †, Saksham Kumar ‡
*ITB, 12213099, Dept of Computer Science, NIT Kurukshetra
Email: 12213099@nitkkr.ac.in
†ITB, 12213117, Dept of Computer Science, NIT Kurukshetra
Email: 12213117@nitkkr.ac.in
‡ITB, 12213103, Dept of Computer Science, NIT Kurukshetra
Email: 12213103@nitkkr.ac.in

***Abstract*—Our project aims to make an AI-powered system that helps to automate and improve the process of resume screening done by all corporations and firms. It tells the compatibility of a candidate by matching his or her resume with the provided job description and classifies an applicant into 3 different categories: no fit, potential and good fit. Moreover, the system calculates the ATS (Applicant Tracking System) score which can be utilized to know how well suited the candidate is for the following job. The project is implemented using NLP (Natural Language Processing) along with machine learning, deep learning and artificial intelligence (AI) techniques to validate key factors such as skills, experience and qualifications provided in the applicant's resume. We can suggest missing keywords from the resume to improve skill gap analysis helping applicants and recruiters to pinpoint scope of improvement. A ranking system further prioritizes resumes based on the calculated ATS scores, smoothening the entire shortlisting and hiring process. Our project's methodology aims to increase recruitment efficiency and promote fairness through intelligent automation techniques.**

***Index Terms*—Resume Screening, Job Matching, NLP, Deep Learning, Skill Gap Analysis, ATS, Ranking System**

I. Introduction

A. Background

With the rapid advancement and modernisation in different sectors of the workforce and economy, the majority of the corporations, organisations and firms have shifted from the conventional hiring process. With the advent of digital hiring platforms and ease of using online applications have led the organisations to make a significant transformation. Companies today receive not only hundred but a few thousand job applications for a single opening, making it extremely difficult to manually screen all the submitted resumes and thus evaluate it on different parameters. Traditional ATS softwares somehow solve this issue, but they do rely on rigid keyword matching, leading to rejection of potentially suitable candidates whose resumes are not in accordance with the above softwares.

B. Problem Statement

There is a serious need for a more suitable and intelligent automation system that can accurately match resumes with their respective job descriptions, not only relying on keyword matches but also by evaluating relevant skills, qualifications and experience. Existing systems lack the understanding of semantic relationships between resumes and job descriptions, leading in inefficient evaluation and poor job matches.

C. Objectives

- To develop a resume-jd matching model that helps in classifying an applicant's resume into : no fit, potential and good fit.
- To calculate a compatibility score, known as ATS (Application Tracking System) score.
- To implement keyword matching which further helps in improving skill gap analysis.
- To prioritise different resumes based on ATS scores helping the recruiters shortlist the suitable candidates quickly, saving a significant portion of time.

D. Motivation

The motivation behind this project is to enhance the accuracy, efficiency, and fairness of the recruitment process by integrating ML, DL and AI into resume screening. A well-designed, intelligent matching system helps not only recruiters—by reducing time and effort—but also job seekers, by improving the visibility of selected candidates and identifying areas for further improvements in resumes.

E. Overview

This report is neatly classified and structured as follows: Section I explains the introduction part which is further classified into: background, problem statement, objectives, motivation and overview of the report. Section II talks about the related work already done in this area of study and research. Section III provides knowledge of the methodologies used including information about the dataset and different ML, DL and AI techniques used in the project. Section IV provides us with the insights of the project's results. Finally, Section V concludes the report.

II. Related Work

Initially basic machine learning classifiers were used to categorise different resumes into different class labels based on resume and jd matching. Post this, more advanced ML techniques including ensemble methods have been utilized

for getting better accuracies. Later, deep learning models specifically RNNs (Recurrent Neural Networks) [1] have come into use due to its ability to work well with textual data. More advanced architectures of RNNs, such as LSTMs (Long Short Term Memory) [2] and GRUs (Gated Recurrent Unit) [3]can be implemented for better results. These models, along with proper usage of DL techniques (batch normalisation, drop outs, early stopping) can severely improve efficiency of the system.

Several studies have explored and utilised natural language processing (NLP) techniques to improve resume parsing and job matching. For generating word embeddings , Word2Vec and GloVe [4]have been used to represent textual content in a dense vector space, paired up with cosine similarity which is used to find the ATS (Applicant Tracking System) score. More recent and advanced approaches make use of transformer-based models like BERT [5] and RoBERTa [6] to understand context means and semantics more effectively.

Research also highlights the importance of prompt engineering (or AI) in this, where models not only rank resumes based on ATS scores but also suggest improvements by providing insights on skill gap analysis and important keywords relevance. Several systems often consider key factors such as education, experience and relevant skills, while some systems attempt to match resumes based on job market trends and historical hiring data of corporations and firms.

III. Methodology Used

This project aims to automate the resume categorization against job postings into three labels: **Good Fit**, **No Fit**, and **Potential Fit**. The methodology used is structured along a typical Natural Language Processing (NLP) pipeline comprising processes such as data preparation, text preprocessing, and the application of machine learning (ML), deep learning, and ensemble methods. We also utilized large language models (LLMs) to generate ATS (Applicant Tracking System) scores.

A. Data Collection and Preprocessing

The data was collected from Hugging Face [7] and had 6,242 rows. It has two columns: *text* and *label*. The *text* column has concatenated resumes and job descriptions separated by a custom delimiter `<<,>>`, and the *label* column indicates the classification: Good Fit (24.7%), No Fit (50.4%), and Potential Fit (24.9%). Labels were translated into numerical values for training.

Preprocessing steps included:

- Splitting the *text* column into independent *resume* and *job description* columns.
- Converting all text into lowercase.
- Deletion of URLs, email addresses, and special characters.
- Removal of duplicate headers such as "Summary" or "Overview".
- Tokenization and padding of sequences of text up to a sequence length of 512 tokens.
- Token encoding with 100-dimensional GloVe embeddings.

B. Machine Learning Approach

Traditional models like Support Vector Machines (SVM) [8], Naive Bayes [9], Random Forest [10], and Logistic Regression [11] were trained with TF-IDF features. But these models found it difficult to understand semantic and contextual meaning and hence had limited performance.

C. Deep Learning Models

To improve results, we employed three architectures with TensorFlow and Keras:

- **BiLSTM-Based Model [12]**: Applied bidirectional LSTM layers on both resume and job description.
- **BiGRU-Based Model** [13]: Used bidirectional GRU layers for computational efficiency.
- **Hybrid Model**: Employed BiLSTM for resumes and BiGRU for job descriptions.

All the models employed GloVe-initialized embeddings for training. They were preceded by dense layers with ReLU activation, dropout regularization, and a softmax classifier. Among these, the best result was obtained from the BiLSTM-based model.

Dropout layers provided regularization advantages and encouraged additional investigation into ensemble techniques.

D. Ensemble Learning

Spurred by the dropout regularization impacts, we investigated ensemble ML methods. Gradient boosting models such as LightGBM [14], XGBoost [15], AdaBoost [21] , Random Forest [10] and CatBoost [16] were utilized because of their capability to tackle feature importance and class imbalance. These models far outperformed conventional ML models.

E. Hyperparameter Tuning

Early ML models such as SVM [8], Naive Bayes [9], and Logistic Regression [11] performed poorly. Hyperparameters were optimized using `RandomizedSearchCV`, which enhanced their evaluation scores. `KerasTuner` was similarly applied to the overperforming BiLSTM model, enhancing its accuracy from 96% to 99%.

Best Hyperparameters for BiLSTM Model:

- `lstm_units`: 32
- `dense_units`: 64
- `dropout1`: 0.3
- `dropout2`: 0.4
- `learning_rate`: 0.001

F. ATS Score Generation

Besides classification, we sought to produce an ATS score for resume-job matching. Early tries with cosine similarity on TF-IDF and embeddings resulted in poor differentiation.

5 transformer models were then used [17]—MPNet [18], MiniLM [19], RoBERTa [6], and BERT [5]—to create contextual embeddings. Cosine similarity was calculated, and statistics such as mean and standard deviation were employed to gauge ATS scores. Label differentiation still proved inadequate.

To enhance this, we employed the Gemini 1.5 Flash API [20] to extract keywords from job descriptions. These keywords were varied and used against resumes using prompt-based LLM queries. This highly enhanced ATS score relevance and differentiation.

IV. Experimental Results

A. Overview

This part introduces a thorough assessment of our suggested resume-job fit classification framework. We set the baseline results utilizing common machine learning classifiers trained on TF-IDF features first. Then, we compare the performance of some deep learning models, such as LSTM [22], Bi-LSTM [12], Bi-GRU [13], and a hybrid Bi-LSTM+Bi-GRU structure with GloVe embeddings [4]. Ensemble models like LightGBM [14], XGBoost [15], and CatBoost [16] are then tested as effective and high-performance alternatives. We also evaluate the performance of our ATS scoring pipeline with both baseline and transformer-based embeddings. We conclude by providing some important observations and comparing our method with other approaches to illustrate the real-world benefits of our system.

B. Baseline Performance: Traditional ML Models

Table summarizes the results of conventional machine learning classifiers trained on TF-IDF features. These models underperformed due to their inability to capture semantic relationships.

TABLE I: Performance of Traditional ML Models

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	66.45%	67.97%	66.45%	64.09%
SVM - Linear	56.52%	54.77%	56.52%	52.94%
SVM - RBF	53.16%	49.95%	53.16%	48.57%
SVM - Poly	51.48%	47.39%	51.48%	47.14%
Naive Bayes	52.04%	49.14%	52.04%	49.65%

C. Deep Learning Results

Figure compares the deep learning architectures evaluated.

- The Bi-LSTM [12]-based model yielded the best results with an overall accuracy of **96.00%**.
- The Bi-GRU [13] and Hybrid (BiLSTM+BiGRU) models followed closely but could not match Bi-LSTM.
- All models used GloVe [4] embeddings and achieved strong contextual understanding.

TABLE II: Accuracy of Deep Learning Models

Model	Accuracy
LSTM	93.31%
Bi-LSTM	96.00%
Bi-GRU	93.00%
Hybrid (Bi-LSTM + Bi-GRU)	94.00%

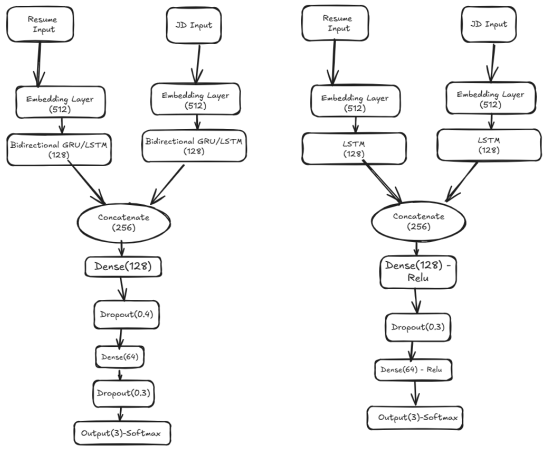

Fig. 1: Architecture of the Deep Neural Network used for Resume Classification. The model utilizes GloVe embeddings followed by stacked bidirectional LSTM/GRU layers, dense layers, and a final softmax output.

D. Ensemble Learning Performance

Ensemble models outperformed traditional classifiers and offered a lightweight alternative to deep learning. Among them:

- **LightGBM [14]** achieved the highest accuracy (**98.56%**) and the best class balance.
- **XGBoost [15]** and **CatBoost [16]** closely followed with competitive precision and recall values.

TABLE III: Performance of Ensemble Learning Models

Model	Accuracy	Precision	Recall	F1-Score
LightGBM	98.56%	98.56%	98.56%	98.56%
XGBoost	98.40%	98.40%	98.40%	98.40%
CatBoost	98.32%	98.32%	98.32%	98.32%
Random Forest	94.56%	94.59%	94.56%	94.53%
AdaBoost	75.66%	76.22%	75.66%	75.29%

E. ATS Score Evaluation

Initial ATS scores based on cosine similarity over TF-IDF and GloVe embeddings lacked meaningful separation among labels. Transformer-based embeddings (e.g., MPNet, RoBERTa) improved this, but the real breakthrough came with:

- Gemini 1.5 Flash API [20] for keyword extraction from job descriptions.
- LLM-based matching between keywords and resumes via prompt engineering.

The updated ATS scoring pipeline displayed a strong correlation with classification labels. Resumes labeled as **Good Fit** consistently scored higher ATS values, while **No Fit** samples showed significantly lower scores.

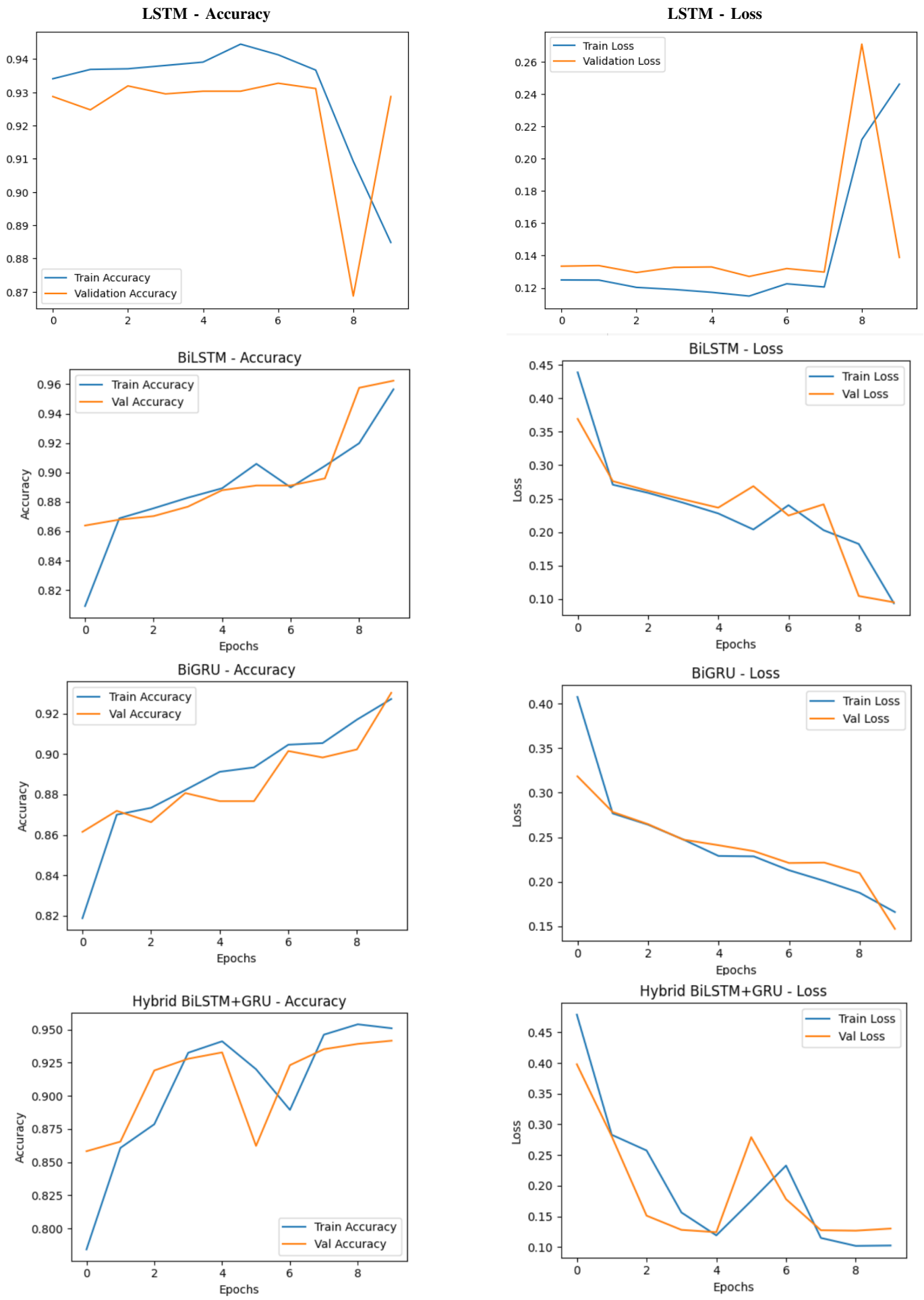

Fig. 2: Training and Validation Accuracy and Loss for Deep Learning Models.

TABLE IV: Mean similarity scores across three class labels for five models

Model	Good Fit	No Fit	Potential Fit
all-mpnet-base-v2	0.6349	0.5627	0.6061
paraphrase-MiniLM-L6-v2	0.5718	0.5247	0.5507
sentence-t5-base	0.6471	0.5867	0.6271
all-roberta-large-v1	0.5966	0.5114	0.5831
bert-base-nli-mean-tokens	0.7691	0.7343	0.7567

TABLE V: Standard deviation of similarity scores across three class labels for five models

Model	Good Fit	No Fit	Potential Fit
all-mpnet-base-v2	0.0967	0.1244	0.1159
paraphrase-MiniLM-L6-v2	0.1124	0.1300	0.1232
sentence-t5-base	0.1019	0.1164	0.1043
all-roberta-large-v1	0.1236	0.1390	0.1286
bert-base-nli-mean-tokens	0.0738	0.0810	0.0798

F. Analysis and Observations

- The BiLSTM [12] model's strong performance highlights the capability of bidirectional long short-term memory networks in effectively capturing contextual dependencies within resume-text data.
- Transformer embeddings were more effective for semantic comparison than static embeddings.
- LLM-based keyword matching not only boosted ATS relevance but also introduced explainability.
- Ensemble models proved to be robust, efficient alternatives for real-time deployment.

G. Comparison with Previous Approaches

Conventional resume screening models have mostly used simple methods such as TF-IDF and keyword detection, which tend to miss the underlying semantic relationship between resumes and job descriptions. Our method, on the other hand, uses a multi-stage pipeline using traditional machine learning models, deep learning models (e.g., BiLSTM [12] and BiGRU [13]), and sophisticated ensemble methods such as LightGBM [14] and XGBoost [15]. In addition, we integrate large language models (LLMs) to produce ATS scores through clever keyword extraction and matching relevant to a job. Such an all-encompassing methodology not only optimizes the process of classification but also offers more understandable and context-sensitive resume ratings.

V. Conclusion

Through this project, we created a smart system for classifying resumes based on job descriptions into three useful categories: Good Fit, Potential Fit, and No Fit. Our methodology combined machine learning in its conventional sense, deep neural networks, ensemble methods, and big language models to deal with the subtlety of mapping between resumes and job postings. Apart from classification, we have also worked towards the generation of ATS scores via both transformer-based and LLM such as Gemini that supports keyword-relevant resume analysis as well as skill mapping, thereby supporting resume screening by relevance of keywords and skill. Our system hopes to make recruitment processing more efficient and objective. In the future, we expect to extend the system to deal with multilingual resumes, incorporate it with real-time job portals, and investigate fairness and bias mitigation techniques to make hiring decisions fair and balanced.

VI. Code and Resources

Code for this project: *Colab Notebook for Resume Classification Using Deep Learning*.
You can see the code here.

References

[1] Zachary Lipton, *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Available at: arxiv.org/abs/1506.00019

[2] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. *A Review on the Long Short-Term Memory Model*. Artificial Intelligence Review, 2020. DOI: 10.1007/s10462-020-09838-1. Available at: arxiv.org/abs/1705.05690

[3] Rahul Dey and Fathi M. Salem. *Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks*. Available at: arxiv.org/abs/1701.05923

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543. 2014. DOI: 10.3115/v1/D14-1162

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Available at: arxiv.org/abs/1810.04805

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Available at: arxiv.org/abs/1907.11692

[7] Apoorv. *Resume-JD Match Dataset*. Available at: huggingface.co/datasets/facehuggerapoorv/resume-jd-match

[8] Cortes, C., Vapnik, V. *Support-Vector Networks*. Machine Learning, 20(3), 273–297. 1995. DOI: 10.1007/BF00994018

[9] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. Proceedings of the Eighteenth International Conference on Machine Learning, 2001. Available at: cs.cmu.edu/ mccallum/bib/papers/crf-lafferty-icml01.pdf

[10] Leo Breiman. *Random Forests*. Machine Learning, 45(1), 5–32. 2001. DOI: 10.1023/A:1010933404324

[11] David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. Wiley Series in Probability and Statistics, 2013. Available at: wiley.com

[12] Mike Schuster and K. K. Paliwal. *Bidirectional Recurrent Neural Networks*. IEEE Transactions on Signal Processing, 45(11), 2673–2681. 1997. DOI: 10.1109/78.650093

[13] Józef Z. S. and S. A. P. *BiGRU: A Bidirectional Gated Recurrent Unit Network for Sequence Modeling*. Available at: arxiv.org/abs/1702.02814

[14] Guolin Ke, Qi Meng, Weidong Dong, Hui Li, and Tie-Yan Liu. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. Advances in Neural Information Processing Systems (NeurIPS), 2017. Available at: arxiv.org/abs/1711.08784

[15] Tianqi Chen and Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. DOI: 10.1145/2939672.2939785

[16] Evgeny S. and Alexey G. *CatBoost: A High-Performance Gradient Boosting on Categorical Features*. Available at: arxiv.org/abs/1706.09516

[17] Amir Adridi. *Job Resume Matching: Models Evaluation*. Available at: github.com/amiradridi/Job-Resume-Matching

[18] Xiaodong Liu, Pengcheng He, Weizhu Chen, Li Dong, and Jianfeng Gao. *MPNet: Masked and Permuted Pretraining for Language Understanding*. In Proceedings of the 37th International Conference on Machine Learning (ICML), 2020. Available at: arxiv.org/abs/2004.09297

[19] Wenxiang Hong, Chenfei Wu, and Yiming Yang. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Pretraining*. In Proceedings of the 38th International Conference on Machine Learning (ICML), 2021. Available at: arxiv.org/abs/2002.10957

[20] Google AI. *Google AI - Advancing the State of the Art in AI*. Available at: ai.google.dev

[21] Yoav Freund and Robert E. Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 55(1), 119–139, 1997. DOI: 10.1006/jcss.1997.1504

[22] Hochreiter, Sepp and Schmidhuber, Jürgen. *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735