Natural Language Processing with deep learning

Muhammad Umer Bin Yaqoob0
dept. Electronic Engineering

***Abstract*—This study presents an in-depth exploration of NLP through the evolution from rule-based systems to advanced deep learning methodologies. It emphasizes the transition from statistical methods to the deployment of neural networks, such as RNNs and LSTM networks, which are proficient in handling sequential data and recognizing temporal and contextual nuances in text. With the advent of attention mechanisms and the Transformer architecture, NLP models have significantly increased in efficiency and efficacy. These days companies receive hundreds of applications against each job opening which adds to the work of sorting through all these resumes to find out candidates that are required for the job. In this research we have proposed a DistilBERT based deep learning classification technique to predict job titles based on the resume data so that companies that need to speed up the recruitment process they don't have to go through a lengthy process of sorting through resume documents. Results suggest that if we have enough dataset to train the model on each job title it can significantly reduce the time taken by this process.**

I. Introduction

Natural Language Processing (NLP) has progressively grappled with the complexities of human language through the lens of computational intelligence. Initially, the field of NLP depended on a foundation of rules crafted by linguists to decode and interpret text, which proved to be fragile and constrained due to the dynamic and intricate nature of language. A pivotal transition occurred with the introduction of statistical methods, where machine learning algorithms began to discern linguistic patterns within extensive text datasets for language prediction. The true metamorphosis, however, was catalyzed by the emergence of deep learning approaches within the field of NLP. The deep learning revolution in NLP was characterized by the deployment of intricate neural networks with multiple layers. This revolution was further propelled by the deployment of architectures such as Recurrent Neural Networks [] (RNNs) and Long Short-Term Memory [2] (LSTM) networks, which were adept at handling data sequences and recognizing temporal relationships and contextual nuances within text. Innovations continued with the advent of attention mechanisms and the transformative Transformer architecture, which facilitated the simultaneous processing of text sequences, thereby markedly enhancing the efficiency and efficacy of NLP models. This gave rise to models like Bidirectional Encoder Representations from Transformers [3] (BERT), Generative Pretrained Transformer [4] (GPT), and its variants [5], which have set new standards in NLP tasks through their ability to understand context and generate human-like text. Despite these advancements, challenges in NLP persist. Ambiguity, context-sensitivity, and the subtleties of human language make NLP a complex field. Deep learning addresses these challenges by leveraging large datasets and computational power to learn nuanced patterns in language. Transfer learning [6] and unsupervised pre-training allow models to develop a broad understanding of language before being fine-tuned on specific tasks, enabling them to handle a wide range of NLP challenges. However, issues such as the need for vast amounts of annotated data, the interpretability of models, and the computational resources required remain areas of active research and development.

II. Background and Related Work

The trajectory of NLP methodologies has been marked by significant milestones, evolving from rule-based systems [7] to advanced deep learning techniques. Initially, NLP systems were constructed around linguistic rules developed by experts, which allowed for the parsing and interpretation of text but lacked scalability and struggled with the nuances of language. The emergence of machine learning in NLP brought statistical models to the forefront, utilizing algorithms like decision trees [8] , Naïve Bayes [9], and Support Vector Machines [10] to learn from data. However, these models were limited in their ability to capture context and sequence information inherent in language. The integration of deep learning into NLP has been transformative, introducing models with the capacity to identify complex patterns within voluminous datasets. RNNs and LSTMs have been pivotal in addressing the sequential nature of text, facilitating the identification of inter-sentence and inter-paragraph dependencies. Convolutional Neural Networks [11] (CNNs), traditionally associated with image processing, have been repurposed for NLP tasks, enabling the detection of textual patterns and the execution of classification tasks. The unveiling of the Transformer architecture, as expounded in the seminal work "Attention is All You Need" [12], constituted a paradigm shift within NLP. This architecture eschewed the sequential processing of RNNs and LSTMs in lieu of attention mechanisms, granting models the ability to appraise the significance of words across a sentence or document, independent of sequential position. This advancement in parallel processing substantially enhanced the model's operational efficiency and effectiveness. BERT further extended

this framework by integrating bidirectional context, allowing for the comprehension of contextual relationships surrounding each token from both directions, thereby achieving significant gains in language processing tasks. The development trajectory of NLP continued with the GPT series, which augmented the Transformer's architecture to emphasize unsupervised learning and the generation of contextually coherent text. Successive iterations of GPT have seen an increase in model size and complexity, resulting in architectures with extensive parameter sets capable of performing a broad spectrum of language-related tasks without the necessity for task-specific pre-training. These developments have been instrumental in shaping the current landscape of NLP research, which is persistently pushing the boundaries of language comprehension and generative capabilities, with the aim of developing more sophisticated AI systems. The field of NLP is characterized by continuous innovation, with research efforts dedicated to addressing the challenges related to model efficiency, interpretability, and the ethical deployment of NLP technologies.

III. Foundations of Deep Learning

Inspired by the neural networks of the human brain, artificial neural networks [13] (ANNs) are structured as a series of nodes, termed "neurons," interconnected by "synapses" that carry signals. Inputs are received by these neurons, processed via an activation function, and the resulting output is relayed forward. The perceptron, a basic ANN form, comprises a singular neuron with tunable weights and bias, and is adept at performing linear classifications. Neurons within more complex, multi-layered networks are organized into distinct layers: the input layer accepts the initial data, hidden layers perform computations, and the output layer delivers the final verdict. The connection strengths, or weights, modulate the impact of these inputs on the outputs, with network training entailing the adjustment of these weights to minimize output errors against expected results. Deep learning, a specialized branch of machine learning, is characterized by its use of multi-layered neural networks, facilitating a tiered approach to feature learning. This stratified model architecture permits the extraction of increasingly complex patterns from basic elements, a concept known as feature hierarchy. Early network layers might identify rudimentary features such as edges in visual data or phonemes in audio processing, while deeper layers synthesize these to discern more intricate forms like shapes or specific lexicon. Unlike traditional machine learning techniques that necessitate handcrafted feature extraction, deep learning architectures inherently and dynamically learn these feature hierarchies from the data. The training of deep neural networks is underpinned by back propagation [14], an algorithm that calculates the gradient of the loss function concerning the network's weights. The network's forward pass circulates the input to produce an output, which is then evaluated against the target output by the loss function. Subsequently, back propagation undertakes a reverse pass to determine the loss function's gradient, informing weight adjustments to reduce the loss, thereby optimizing the network's performance.

IV. Model-Based Design for NLP

Model-based design in the context of NLP adheres to a systematic approach to building models that are capable of understanding, interpreting, and generating human language. The principle revolves around creating a mathematical and computational representation of linguistic phenomena that can be trained to perform specific tasks such as translation, sentiment analysis, or entity recognition. This design philosophy emphasizes the importance of incorporating linguistic theory, cognitive insights, and statistical patterns derived from language data into the model architecture. It often involves selecting or crafting a model structure that aligns with the linguistic properties of the task, such as sequence models for tasks with a strong temporal component or attention mechanisms for tasks requiring contextual awareness.

The initial stage in an NLP workflow is data preprocessing, essential for converting unstructured text into a format that is amenable to machine learning algorithms. This stage encompasses a variety of processes such as breaking down text into tokens, standardizing text, reducing words to their root forms, and extracting the base form of words, as well as eliminating common words and extraneous data. Subsequently, the cleansed data is transformed into a numerical form, typically vectors, using methods such as binary representation, TF-IDF [15], or contextualized word representations. The subsequent phase is the construction of the model, which entails selecting a neural network structure that is congruent with the specific NLP task, ranging from traditional sequential models like RNNs and LSTMs to advanced Transformer models for tasks that demand an understanding of extensive contextual dependencies. This stage also involves determining the architecture specifics, such as the number of processing layers, network dimensions, neuron activation mechanisms, and the patterns of inter-neuron connections.

In the concluding evaluation phase, the model's performance is gauged using metrics tailored to the NLP task, such as accuracy, precision, recall, and F1 score for classification tasks. The model's performance is evaluated on a separate test set that was not seen during the training phase to ensure that the model generalizes well to new data. This phase may also involve error analysis and the refinement of the model based on its performance, leading to iterative improvements in the design and training process.

V. Implementation of Deep Learning Algorithm for NLP

Companies often get thousands of resumes for a single job opening and have specialized staff to sift through them to find eligible candidates. Finding the right employees is tough for any business, and it's even harder for those in industries that need a lot of workers, are growing fast, or see a lot of people leaving their jobs. In the competitive IT sector, there's a constant need for skilled professionals with diverse

technical abilities and knowledge of different business areas. These professionals are recruited and placed on projects to tackle client problems. In this case study we have employed a deep learning technique to screen resumes without human intervention. The steps involved in it are as follows:

A. Dataset

' This study utilized a dataset comprised of resumes [16], each tagged with a 'Category' field indicative of the job classification. Initial analysis focused on comprehending the dataset's structure. Identifying the range of job categories within the 'Category' field constituted the primary objective, providing insight into the dataset's diversity.

B. Data Preprocessing:

Prior to initiating the modeling stage, it was imperative to refine and structure the dataset. The overarching aim was to enhance the clarity and analyzability of the resume text. Several critical steps were involved in this cleaning process:

- An examination of the text was conducted to identify and discard unnecessary elements for resume categorization, such as URLs, hashtags, and mentions. These elements were eliminated due to their irrelevance in assessing an individual's qualifications and professional background. They are often used in social media or web contexts and are not relevant to the content of a resume. By removing such information, the focus is kept on the substantive content that is indicative of a candidate's background and skills.
- All punctuation marks and special characters were removed to reduce complexity and ensure that the analysis focuses on the actual words and phrases in the resumes. Simplifying the text in this manner aids in creating a more streamlined dataset, making it easier for the model to identify patterns and relationships.
- Non-ASCII characters were removed to maintain consistency and standardization across the dataset. These characters can introduce variability and may not be interpreted uniformly during the classification task. By ensuring that only standard ASCII characters are used, the dataset becomes more accessible and less prone to errors during further tasks.
- Extra spaces were reduced to single spaces to maintain uniformity in the textual data. Inconsistent spacing can lead to discrepancies and may cause similar words or phrases to be treated as distinct entities, thereby affecting the accuracy of the analysis. By standardizing spacing, the dataset becomes more coherent and easier to process.
- Stop words were removed because these are the most common words such as "the", "is", "in", and "and" that occur frequently in text but usually don't carry significant meaning on their own. By removing stop words, the analysis can focus on words that are more likely to be indicative of a candidate's skills and experiences.
- Punctuation marks were removed because these can introduce additional complexity in the text, making it harder to extract meaningful patterns. By removing punctuation, the text is simplified, allowing for a more straightforward analysis.

C. Text Analysis

Various text analysis techniques were employed to gain insight into the dataset. Details of these techniques are as following:

- A bar plot was used to visualize the number of resumes corresponding to each category. This approach helps in understanding the distribution of resumes across different job categories, highlighting areas with higher or lower representation.
- A pie chart showcased the distribution of the categories within the dataset. This visual representation allows for a quick and easy understanding of how the various categories compare in terms of their proportion within the dataset.
- Frequency distribution of words was plotted because it is crucial for understanding common themes, assessing data quality, and aiding in feature selection for machine learning models. It helps identify frequently occurring words, revealing patterns, trends, and potential issues in the data.
- A word cloud was generated because it is a useful method for visually summarizing textual data. This allows for the quick identification of key themes and prominent terms, such as skills or job roles, which are crucial for understanding the focus areas of the resumes.

D. Tokenization

Tokenization is a fundamental step in natural language processing that involves transforming a sequence of text into individual elements, or tokens, which can be words, subwords, characters, or phrases. This process is crucial for several reasons. Firstly, it simplifies text processing by breaking down complex textual data into more manageable and analyzable pieces. Secondly, it facilitates text analysis as algorithms can process and comprehend data more effectively when presented as tokens.

In our work we have used BERT tokenizer. It is designed to function optimally with the BERT model and employs WordPiece tokenization. This approach segments words into smaller subwords or characters, particularly when the words are not present in its predefined vocabulary. This feature is crucial as it enables the BERT tokenizer to effectively manage out-of-vocabulary words by decomposing them into identifiable components.

Furthermore, the tokenizer is engineered to operate synergistically with BERT's bidirectional context, enhancing the nuanced comprehension of each word within its contextual framework. The BERT tokenizer is employed in classification tasks for various reasons, including its integration with a pre-trained model and tokenizer, both of which are refined through extensive training on a large text corpus, equipping it with an extensive grasp of language. This pre-training ensures

efficiency and alignment with the model's expectations, often resulting in enhanced classification results.

E. Classification

BERT is a state-of-the-art pre-trained deep learning model designed for natural language processing tasks. It is unique in its use of bidirectional context, meaning it considers both the preceding and following words in a sentence to understand the context and semantics of a particular word. It is highly effective for text classification tasks due to its ability to capture complex language nuances and contextual information. By leveraging pre-trained knowledge from large corpora, it can be fine-tuned on a specific classification task, often requiring less training data and time while achieving superior performance compared to models trained from scratch. In this study we have used DistilBERT [17], which is a compact variant of the original BERT model, designed to be smaller, faster, and more efficient while still capturing the essence of BERT's capabilities. The "distil" in its name signifies that it has been distilled from BERT, meaning it has undergone a process where a smaller "student" model learns to replicate the performance of the larger "teacher" model. The "base" indicates that it's a standard size model, not as large as some of the bigger BERT variants, but still powerful enough for a wide range of tasks. Despite its reduced size, DistilBERT manages to retain a significant portion of the original BERT's performance.

In the classification process, the model underwent a series of training and validation loops across 4 epochs. During the training phase, the model, set in training mode, iterated over batches of data, computing the loss based on its predictions and the actual labels. To enhance the model's ability to generalize and to mitigate the risk of overfitting, the loss function was augmented with L1 and L2 regularization components. The L1 regularization promoted a sparse model representation by imposing a penalty based on the absolute values of the parameters, whereas L2 regularization constrained the magnitude of individual weights by penalizing the square of their values. These regularization methods are crucial, as they direct the model to prioritize significant features and disregard irrelevant data variations. Subsequently, gradients were calculated and applied to refine the model's parameters. In parallel, the model underwent evaluation against a distinct validation dataset to gauge its ability to generalize. This validation involved generating predictions, quantifying the loss, and measuring performance metrics such as accuracy, precision, recall, and F1-score. Following each training epoch, a comprehensive report detailing both training and validation outcomes, as well as the effects of regularization, was compiled to track the model's progress. The validation phase, in concert with L1 and L2 regularization, played a pivotal role by providing insight into the model's performance on novel data, thus enabling the identification and rectification of overfitting or underfitting.

The dataset was segmented into three subsets: 80% allocated for training purposes, 10% reserved for validation to monitor the learning progress, and the remaining 10% set aside for final testing on new, unseen data. This distribution ensured that while the model had ample data for learning, a sufficient portion was retained to verify the learning integrity and to evaluate the model's final performance. During the testing phase, the model was evaluated on a separate dataset to assess its generalization capabilities. The model, previously set to evaluation mode, processed the test data in batches, ensuring that operations such as dropout and batch normalization were appropriately adjusted for the inference context. For each batch, predictions were generated, and a loss metric was computed to quantify the model's performance. These predictions were then collated to form a comprehensive picture of the model's accuracy across the entire test set. The whole process of classification is presented in Figure 1.

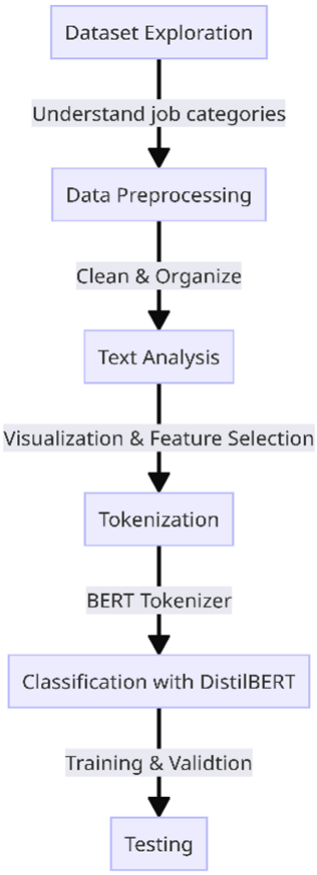

Fig. 1. Classification Process

VI. Evaluation Matrices

The model was evaluated using several standard metrics that are commonly used to assess the performance of classification models. These metrics include:

- **Accuracy:** This is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. It is suitable when the class distribution is similar. However, accuracy alone can be misleading if there is class imbalance in the dataset.

- **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to a low rate of false positives and is a measure of a classifier's exactness. High precision indicates a low number of false positives.
- **Recall:** Recall is the ratio of correctly predicted positive observations to all observations in the actual class. It is a measure of a classifier's completeness. High recall indicates a low number of false negatives.
- **F1-Score:** The F1 Score is calculated as 2*((precision*recall)/(precision+recall)). It is also known as the F-Score or F-Measure. The F1 Score is the harmonic mean of precision and recall, reaching its best value at 1 and worst at 0.
- **Loss:** This measures how well the model's predicted probabilities match the actual class labels. It is often used during training to perform gradient descent and update the model's weights. Lower loss values indicate a better model.

For each epoch, these metrics are calculated for both the training and validation datasets to monitor the model's learning progress and its performance on unseen data. After training, the same metrics are evaluated on a separate test dataset to assess the model's generalization capability.

The classification report provides these metrics for each class, as well as macro and weighted averages:

- **Macro Average:** This is the average of the metric for each class, without taking the class imbalance into account.
- **Weighted Average:** This takes the class imbalance into account by weighting the metric by the number of true instances for each class.

These metrics together provide a comprehensive view of the model's performance and are particularly useful for identifying specific areas where the model is performing well or where it may need improvement.

VII. Results

In this dataset Java Developer roles are the most common, with 84 instances, followed by Testing roles at 70. DevOps Engineer and Python Developer roles are also frequent, with 55 and 48 counts respectively. Web Designing, HR, and Hadoop related roles are represented with a similar frequency, each hovering around the mid-40s. Sales, Mechanical Engineer, Blockchain, Data Science, ETL Developer, and Operations Manager categories each have 40 entries, indicating a balanced demand for these positions. The Arts sector has a slightly lower representation with 36 instances. Database-related roles count 33, while Health and fitness, Electrical Engineering, and PMO roles are present 30 times each. DotNet Developer and Business Analyst roles are noted 28 times, followed by Automation Testing at 26. Network Security Engineer and Civil Engineer roles are slightly less common, with 25 and 24 counts respectively, and SAP Developer roles match the latter. The least represented in this dataset are Advocate roles, with 20 instances, Figure represents these results, and their percentage is presented in Figures.

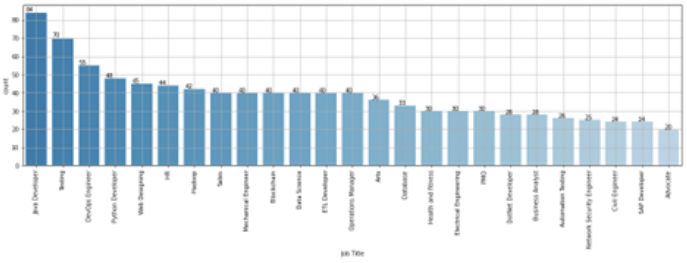

Fig. 2. Jobs Category Count

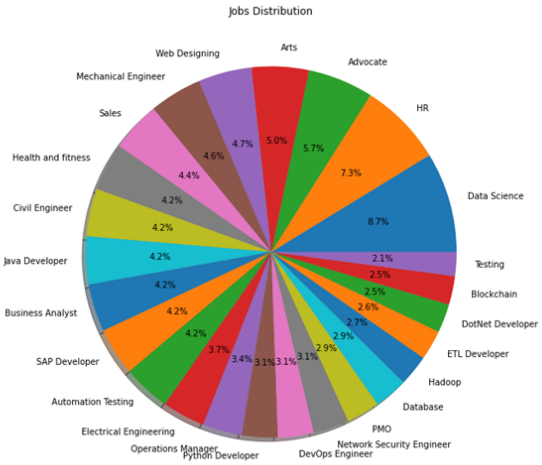

Fig. 3. Jobs Distribution

The Figure shows most common words that appear in resumes, which gives insights into the most prevalent terms used by job applicants. For instance, words like "Experience," "months," and "company" are highly frequent, suggesting that many candidates emphasize their work duration and the organizations they've been associated with. Other common terms include "Project," "data," and "team," indicating a focus on collaborative work experiences and technical skills.

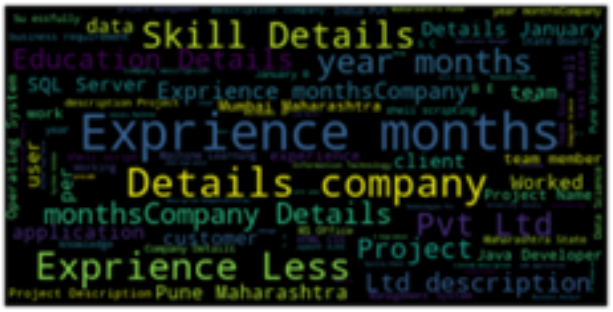

Fig. 4. Word Cloud

The model's learning trajectory over four epochs revealed a progressive enhancement in classification performance. Initially, the model exhibited a modest training accuracy of 7.93% percent, which incrementally improved to 43.56% by the final epoch. Correspondingly, the validation accuracy also saw an upward trend, starting at 10.42% and culminating at 50%. This improvement was mirrored across other key metrics, including precision, recall, and F1-score, all of which are critical indicators of the model's predictive reliability and balance between sensitivity and specificity. Metrics such as precision, gauging the model's accuracy in prediction, and recall, measuring its capacity to identify relevant instances, both exhibited marked improvements. These enhancements signal the model's refined proficiency in pinpointing true positives with a concurrent reduction in both false positives and negatives. The F1-score, which amalgamates precision and recall into a single metric, mirrored this upward trend, indicating a consistent enhancement across these performance measures. In the testing phase that followed the model's training, it registered an accuracy of 55.67%. The precision during testing, at 49.60%, was somewhat below the accuracy, suggesting a higher occurrence of false positives than ideal. The recall was on par with the accuracy, and the F1-score was recorded at 47.73%, which, despite not being exemplary, still denotes a commendable equilibrium between precision and recall, considering the task's intricacy. The detailed classification report delineates the model's performance across different categories, revealing high accuracy in certain categories while highlighting a complete miss in others, as evidenced by scores of zero.This difference highlights how tough it can be for a model to classify many different types of data, especially when some types are not as common as others. The overall scores, which are in the 40-50 percent range, whether they consider the unevenness of the data or not, point out that the model does well with some categories but still needs work to better identify the less common ones.

VIII. Conclusion

The research effectively demonstrates the application of a DistilBERT-based deep learning classification technique in automating the resume sorting process for job title prediction. By leveraging the power of NLP and the efficiency of transformer models, the study successfully illustrates a significant reduction in the time required for sifting through numerous job applications. The accuracy and reliability of the DistilBERT model in distinguishing the subtle differences across a wide range of job titles demonstrate its practicality in the real-world recruitment scenario, offering a valuable tool for companies aiming to optimize their hiring process. In the future, we plan to make the model better by doing two things. First, we will use more resumes from different kinds of jobs to make sure the model works well for a variety of job titles. Second, we will test the model with companies from different types of industries to make sure it's useful no matter what kind of job it's sorting applications for. These steps will help improve how the model works and make it more helpful for companies looking to hire the right people.

References

[1] L. R. Medsker and L. Jain, "Recurrent neural networks," Des. Appl., vol. 5, no. 64–67, p. 2, 2001. This reference discusses Recurrent Neural Networks (RNNs) which are fundamental to deep learning in NLP, particularly for their ability to handle sequential data such as text.

[2] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," ArXiv Prepr. ArXiv160106733, 2016.

[3] S. Alaparthi and M. Mishra, "Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey," ArXiv Prepr. ArXiv200701127, 2020.

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, and others, "Improving language understanding by generative pre-training," 2018.

[5] K. Roumeliotis and N. Tselikas, "ChatGPT and Open-AI Models: A Preliminary Review," Future Internet, vol. 15, no. 6, p. 192, 2023, doi: 10.3390/fi15060192.

[6] L. Torrey and J. Shavlik, "Transfer learning," in Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI global, 2010, pp. 242–264.

[7] M. Santaholma, "Grammar sharing techniques for rule-based multilingual NLP systems," in Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007), 2007, pp. 253–260.

[8] L. Rokach and O. Maimon, "Decision trees," Data Min. Knowl. Discov. Handb., pp. 165–192, 2005.

[9] K. P. Murphy and others, "Naive bayes classifiers," Univ. Br. Columbia, vol. 18, no. 60, pp. 1–8, 2006.

[10] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," IEEE Intell. Syst. Their Appl., vol. 13, no. 4, pp. 18–28, 1998.

[11] P. Kim and P. Kim, "Convolutional neural network," MATLAB Deep Learn. Mach. Learn. Neural Netw. Artif. Intell., pp. 121–147, 2017.

[12] A. Vaswani et al., "Attention is all you need," Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[13] A. Abraham, "Artificial neural networks," Handb. Meas. Syst. Des., 2005.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, pp. 533–536, 1986.

[15] J. Ramos and others, "Using tf-idf to determine word relevance in document queries," in Proceedings of the first instructional conference on machine learning, Citeseer, 2003, pp. 29–48.

[16] S. Anbhawal, "Resume Dataset." 2023. [Online]. Available: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," ArXiv Prepr. ArXiv191001108, 2019.