Received: 14 August 2024 | Revised: 29 October 2024 | Accepted: 16 December 2024 | Published online: 1 April 2025

RESEARCH ARTICLE

Artificial Intelligence and Applications
2025, Vol. 00(00) 1–16
DOI: 10.47852/bonviewAIA52024083

BON VIEW PUBLISHING

Enhancing Recruiter Outreach: Predicting Job Ad Quality with Advanced Machine Learning and NLP Models

Shaida Muhammad[1,*]

[1]*Department of Computing, National University of Sciences & Technology (NUST), Pakistan*

Abstract: In today's competitive recruitment landscape, crafting impactful job outreach messages is essential for attracting top talent. This study presents a novel machine learning and NLP-driven framework for predicting recruiter message quality on professional platforms like LinkedIn, aiming to enhance response rates and hiring success. Our approach leverages a multi-label text classification framework that identifies five critical message attributes: call to action, common ground, credibility, incentives, and personalization. Using a labeled dataset of 97,710 messages annotated across these five categories, we benchmark various machine learning and deep learning models, including Decision Trees, Linear SVM, Logistic Regression, Random Forest, LSTM, and customized transformer-based BERT models. The dataset was meticulously curated to address generalization challenges, with 94,010 samples for training and 3,700 samples in a diversified test set. Model performance was assessed using accuracy, with the customized BERT model achieving 95.67%. Our findings underscore the potential of this framework to enhance recruiter outreach strategies, providing actionable insights to refine message quality and improve candidate engagement.

Keywords: text classification, transformers, deep learning algorithms, machine learning algorithms

1. Introduction

Recruiters in companies use social and professional networks to find candidates for vacant job positions, with LinkedIn serving as the leading professional network, utilized by over 830 million members worldwide. Typically, recruiters on LinkedIn reach out to prospects (potential candidates for the job) through direct messages containing job advertisements. However, the response to these messages varies —prospects may respond with interest, decline, or not respond at all. Ideally, recruiters seek maximum engagement from prospects to increase interview rates and improve the chances of finding the best candidates. For messages to be effective in capturing attention, they need to be strategically crafted with qualities that make them compelling and relevant.

While previous studies in recruitment outreach have mainly focused on general features of persuasive language or single-label classifications, these approaches lack depth in analyzing the specific, multi-dimensional aspects essential for high-quality recruitment messages on professional platforms like LinkedIn. To address this, recruitment experts have identified five key attributes that significantly influence response rates: call to action, common ground, credibility, incentives, and personalization.

Call to action draws immediate attention through prompts like calendar links or phrases like "Let's connect". Common ground builds rapport by emphasizing shared experiences, such as past employment at the same company or similar educational backgrounds. Credibility helps establish trust by highlighting the recruiter's background or company's reputation. Incentives underscore job benefits, attracting interest from prospects. Finally, personalization makes the message feel relevant and individualized, often by using the prospect's name or referencing their work history. A well-structured message ideally incorporates all these elements, which has not been systematically examined in prior research, leaving a gap in understanding their collective impact on response rates.

This research bridges the gap by treating the problem as a multi-label classification challenge. It seeks to determine whether a single message can exhibit multiple persuasive qualities at once. The approach incorporates various techniques from natural language processing (NLP) and machine learning, utilizing both conventional classifiers and advanced deep learning methods, such as a customized BERT model. These methods evaluate message quality across five key dimensions. The findings offer valuable insights for recruiters, helping them refine their strategies and improve engagement with prospective candidates.

2. Related Work

Message classification is a classification task where the text is classified into 5 different categories: call to action, common ground, establish credibility, incentive for connecting, and personalization. Various machine learning and deep learning-based approaches are applied to solve text classification problems in different domains. Machine learning-based text classification consists of two steps: Feature Extraction and Classification Algorithm.

*Corresponding author: Shaida Muhammad, Department of Computing, National University of Sciences & Technology (NUST), Pakistan. Email: smuhammad.mscs19seecs@seecs.edu.pk

2.1. Feature extraction

Natural language is highly unstructured. It is therefore converted to a numerical representation beforehand so that it can be fed to the algorithm. The numerical representation is known as a feature vector or word vector. Different techniques are used in literature for generating feature vectors. The most used are One-hot encoding, Bag-of-Words (BoW), term frequency-inverse document frequency (TF-IDF), etc.

One-hot encoding of words into feature vectors is the simplest approach. A dictionary of size V is created for the text dataset where each word v in the dictionary is mapped to an index number. Feature vectors of size V are created for all words in the vocabulary. Each one-hot vector contains zeros in all positions except for the index position for the respective word. The problem with one-hot encoding is that the feature vector does not carry any meaningful information about the word itself or its context. The distance between words cannot be calculated to show the relationship between words. The size of the one-hot vector is directly proportional to the size of the vocabulary. The one-hot encoding vectors are very sparse, which makes them memory inefficient.

To address the sparsity issue, researchers developed the word co-occurrence matrix approach also referred to as BoW. This method creates a matrix of dimensions $|V| \times |V|$ (where V represents vocabulary size) containing word vectors. These vectors are constructed by examining surrounding words within a defined context window, tallying the frequency of nearby words, and recording these values in corresponding matrix positions. The semantic richness of these feature vectors increases with larger context window sizes. While these vectors maintain the same dimensionality as one-hot encoded vectors, various dimensionality reduction techniques can be applied to compress them, including principal component analysis, singular value decomposition, linear discriminant analysis (LDA), and t-distributed stochastic neighbor embedding. A limitation of co-occurrence matrices is that they capture local contextual relationships but fail to incorporate global text document information.

(TF-IDF) [1] emerged as an alternative methodology that effectively balances both local and global contextual information of words. This technique functions through a two-stage process combining term frequency (TF) [2] with inverse document frequency (IDF) [3]. The TF component quantifies a word's occurrence across documents, generating a matrix with documents as columns and vocabulary terms as rows. For normalization purposes, the standard calculation is adjusted to log(1+TF). Meanwhile, the IDF component is expressed as log(N/df_t), where N represents the total document count and df_t indicates how many documents contain term t. This IDF factor emphasizes rare words that potentially carry greater significance. The final word vector representation is produced by multiplying these two components (TF × IDF), creating a comprehensive representation that balances word frequency with contextual importance

Other deep learning-based approaches like Word2Vec [4], GloVE [5], Doc2Vec [6], etc., are used for feature extraction. They are discussed in detail in subsection 2.3.

Different feature extraction systems and algorithms were used by authors in different domains. The authors [7] used different statistical approaches like filter-based, wrapper-based, embedded feature selection, evolutionary feature selection, LDA, etc., in the domain in medical field for cardiovascular disease (CVD) prediction. The MFFS algorithm was used [8] for selecting important feature in the area of genetic research. MFFS uses firefly algorithm along with deep learning-based approach to extract the best feature for predictions

2.2. Classification algorithms

Classification algorithms take labeled data as input and train a model on it. Training data are represented by $X = \{x_1, x_2, x_3, \ldots, x_m\}$ and $y = \{y_1, y_2, y_3, \ldots, y_m\}$ where X is the set of training examples and y is the corresponding set of labels. x_i is the ith training example, which is an $n-$dimensional vector, and y_i is the label of the ith training example. The labels belong to predefined k classes $\{1, 2, 3, \ldots, k\}$. The machine learning algorithm tries to approximate a function $f(x)$ which labels new unseen data.

In text classification, training data are a set of documents $X = \{d_1, d_2, d_3, \ldots, d_m\}$ where each document d consists of l words, i.e., $d = \{w_1, w_2, w_3, \ldots, w_l\}$. The word vectors are obtained by applying a feature extraction algorithm. The word vectors are summed or averaged to obtain the document vector. Document-level vectors can also be obtained by applying document-level feature extraction algorithms like Doc2Vec. After generating feature vectors from text documents, a classification algorithm is applied to feature vectors to predict the class/labels of the text document.

Logistic regression (LR) is one of the simplest classification algorithms. LR is a modified version of linear regression. Linear regression hypothesizes that class predictions are the weighted sum of the components of the feature vector:

$$\widehat{y^{(i)}} = \theta_0 + \theta_1 x_1^{(i)} + \ldots + \theta_l x^{(i)} \tag{1}$$

LR uses the logistic function:

$$\text{logistic}(\eta) = \frac{1}{1 + \exp(-\eta)} \tag{2}$$

which is a function of the output of linear regression:

$$P\left(y^{(i)} = 1\right) = \frac{1}{1 + exp\left(-\left(\theta_0 + \theta_1 x_1^{(i)} + \ldots + \theta_l x_l^{(i)}\right)\right)} \tag{3}$$

LR gives the probability score for every instance x_i. LR is trained by minimizing the negative log likelihood loss function over training examples m:

$$J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} -y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \tag{4}$$

LR offers significant advantages in computational efficiency, with rapid training and inference times. Nevertheless, it faces certain constraints in practical applications. LR exhibits a tendency to overfit when feature dimensionality exceeds the available training examples. Additionally, its linear decision boundaries present a fundamental limitation when modeling inherently non-linear real-world data distributions, which typically require more complex non-linear functions for accurate representation. Despite these limitations, LR has demonstrated remarkable utility across diverse text classification challenges. Applications include toxic comment detection [9], tweet categorization [10] where LR surpassed alternative methods, comprehensive toxic content classification [11] with LR outperforming competing algorithms across all evaluation categories, emotion analysis in text [12], intent recognition for conversational systems [13], and hierarchical classification of Pakistani news content [14], achieving 83% accuracy in this domain.

Naïve Bayes (NB) is another ML algorithm used for classification tasks. NB uses Bayes' Theorem [15], formulated by Thomas Bayes between 1701 and 1761:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{5}$$

NB makes the assumption that features are independent of each other, which makes it simpler and more efficient to train. Given a document d, the NB classifies the document into a set of classes $C = \{1, 2, 3, \ldots, c\}$ and returns the predicted class $\hat{c}$:

$$\hat{c} = \arg\max_{c \in C} P(d|c)P(c) \tag{6}$$

NB algorithms are used in many areas of text classification, including spam email detection [16] with 99% *F*1-score and thesis topic classification using the abstract [17] where it achieved 88.69% accuracy, 89.76% precision, and 90.49% sensitivity.

One of the simplest and well-known multi-class non-parametric classification algorithms is K-Nearest Neighbors (KNN). KNN assumes that an unseen example x can be classified by looking at the feature space of training data. The class labels of k nearest data points in the training data will decide the class label of x. The distance between x and a training example d can be measured via dot product $Dot(x, d) = \sum_i x_i d_i$, cosine similarity $Cosine(x, d) = \frac{x \cdot d}{|x||d|}$, Euclidean distance $\sqrt{\sum_{i=1}^{n} (x_i - d_i)^2}$, or any other vector similarity metric:

$$f(x) = \arg\max_j Score(x, C_j)$$
$$= \sum_{d_i \in KNN} sim(x, d_i) y(d_i, C_j) \tag{7}$$

$f(x)$ is used to predict the class label of a new example. KNN requires no training time as it is a lazy learning algorithm. KNN does not learn any mapping function during training; the training data are stored in memory, and computations are performed only at inference time. New labeled data points can be added to the dataset anytime to improve accuracy. However, KNN has some limitations. The inference time of KNN is directly proportional to the size of the dataset (i.e., number of data points) and the number of dimensions. If the dataset size is large, it would take a lot of time and computational resources for inference, making it impractical in real-world scenarios. Feature vectors need to be scaled to produce more accurate results. KNN is sensitive to noisy data, so training data must be cleaned before use. Also, selecting the value of k can be challenging. KNN is used for Bengali article classification [18] along with count vectorizer, BBC news text classification [19] where it achieved 92% accuracy, and short text classification [20] where KNN performed better in average f-1 score in one experiment.

Support vector machine (SVM) [21] is a non-linear binary classification algorithm that works well when data is high-dimensional. SVM performs classification by finding a hyperplane in an $n-$dimensional feature space. There could be many hyperplanes between two classes in the feature space. Even though all hyperplanes can classify training data, SVM chooses the one that is best suited for unseen data. SVM selects the hyperplane that has the maximum margin distance. Data points on different sides of the hyperplane are classified into their respective classes, i.e., 1 or 0. The key aspect of SVM is support vectors. Support vectors are used to identify the optimal hyperplane by maximizing the margin distance. SVM uses the hinge loss function: $hingeloss(y, f(x)) = \max(0, 1 - y \cdot f(x))$ to compute the cost and optimize the model weights. SVM has been applied in various domains, including news classification [22], English text classification [23], sentiment analysis [24], news classification using headlines [25], and fake news detection [26]. SVM can also be applied to non-linear data by using a kernel function, which transforms $n-$dimensional non-linearly separable data into $m-$dimensional linearly separable data.

Decision Tree (DT) [27] is another well-known non-parametric classification algorithm. DT learns understandable decision rules from features of the training data by recursively splitting the features. Unlike SVM, KNN, and LR, DT also works on categorical features. DT starts constructing the tree by selecting a feature as the root node and then drawing other nodes from it. DT can be used for multi-label/multi-output classification problems. DT does not require feature scaling and is fast to compute. However, DT can overfit easily, which can be mitigated by using pruning or selecting the tree depth in advance. DT is also very sensitive to noisy data. The accuracy of DT can be improved by using the ensemble technique Random Forest (RF) [28]. RF generates multiple t random trees in parallel. Each tree predicts a label for x, and the final label $\hat{y}$ is decided based on votes. The number of trees influences the accuracy of the model, and the number of trees is directly proportional to the inference time. Therefore, the number of trees should be chosen wisely. DTs have been applied to text data for classification tasks, including product-review sentiment classification [29], Arabic text classification [30], sentiment classification of Roman-Urdu [31], and restaurant review analysis [32].

A study [33] addressed the challenge of cyberbullying on social media by developing a transformer network-based approach combined with machine learning-based algorithm. The method utilized RoBERTa for generating word embeddings and Light Gradient Boosting Machine as the classifier. This approach outperformed traditional machine learning algorithms, such as LR and SVMs. Unlike traditional context-independent word embeddings, this method focuses solely on text data, with potential for future extension to multimedia environments like images, audio, and video.

2.3. Deep learning-based approaches

Machine learning-based text classification approaches were considered quite successful before the arrival of neural networks. Feed-forward neural networks were used as feature extractors to obtain numerical representations of text, where each word in the text gets a vector representation with these models.

Word2vec [4], introduced in, uses two approaches—CBOW (Continuous Bag-of-Words) and Skip-gram to calculate word vectors or embeddings. Word2vec is a two-layer feed-forward neural network where the input is one-hot encoding of a vocabulary of size V. In CBOW, given a context window of a pre-defined size, the model predicts a word inside the context window. For example, with a context window of size two, the input consists of two past and two future words $[W_{t-2}, W_{t-1}, W_{t+1}, W_{t+2}]$, and the model predicts the center word W_t. The input to the model is the average of one-hot vectors of context words. This feed-forward network is trained to maximize the likelihood of the center word given the context window C:

$$\arg\max_{\theta} P(W_{center}/C; \theta) \tag{8}$$

In Skip-gram, the model predicts the context words given the center word W_t. For a context window size of two, the model predicts $[W_{t-2}, W_{t-1}, W_{t+1}, W_{t+2}]$. The softmax function is applied to the final layer to convert raw scores into probabilities, and a simple loss function $y_{pred} - y_{true}$ is used to compute the loss for each context word. The losses are summed to compute the overall cost. The

Skip-gram network is trained to maximize the likelihood of context words given the center word:

$$\arg\max_{\theta} \mathrm{P}\left(\mathrm{C}/\mathrm{W}_{\mathrm{center}};\theta\right) \tag{9}$$

As a two-layer neural network, these models have two weight matrices, W_1 and W_2. These matrices are concatenated $[\mathrm{W}_1\,\mathrm{W}_2]$ and used as word embeddings. These models generate word embeddings such that semantically closer words are closer to each other in $\mathrm{n}-$ dimensional space. The authors of [4] trained word vectors on the Google News corpus, which contains about 6 billion tokens, and evaluated the model on the Semantic-Syntactic Word Relationship test set. The authors found that accuracy is directly proportional to the dimensions of word vectors and the size of the dataset. Higher-dimensional vectors trained on larger datasets tend to produce better results. Word2vec models outperform previous word-embedding models trained on the same dataset with 640 dimensions, and they take local statistics into account for training word embeddings.

The GloVe [5] model incorporates both global and local statistics to generate word embeddings. The GloVe model was trained on a massive corpus, including Common Crawl, Wikipedia dump, and Gigaword5, with a total of 42 billion tokens and a vocabulary size of 400,000. In GloVe, a co-occurrence matrix X is created with a context window of size 10. GloVe provides two sets of word embeddings W and $\widetilde{\mathrm{W}}$ that are added to get the final word vectors. GloVe with different embedding dimensions [34] was used for classification on Google News dataset and word similarity tasks with various datasets and word-embedding sizes. The performance of GloVe was compared on named entity recognition (NER) tasks with other models [35], where it outperformed other models that were not using GloVe embeddings.

These pre-trained word embeddings are used to generate features for different models. The Deep Average Network (DAN) [36] utilizes 300-dimensional GloVe pre-trained word vectors. DAN performs sentiment analysis by taking a text document as input, tokenizing it, averaging the embeddings of all tokens in the text, and then feeding the average embeddings to a simple feed-forward neural network with multiple hidden layers for classification. DAN is efficient in terms of training time per epoch. It achieves comparable accuracy to other sophisticated models like Neural Bag-of-Words (NBOW), RecRNN, and Tree-LSTM, which consider the syntactic information of text and require more training time.

The Doc2vec [6] word-embedding framework was proposed to learn embeddings at the sentence/document level rather than the word level. The architecture of Doc2vec is almost identical to Word2vec, where the model predicts the center word given a context. Doc2vec takes the entire document or paragraph as input and generates sentence-level embeddings. Two variations of Doc2vec were trained: Distributed Memory (PV-DM) and Distributed Bag-Of-Words (PV-DBOW). The PV-DM model is an extension of CBOW. PV-DM takes some random words from a document, treats them as context, and tries to learn the center word of those random words. The context word vectors are averaged or concatenated and provided as input to the model, along with a unique paragraph/document ID vector. The PV-DBOW model is similar to the Skip-gram model. It uses the document ID as input and tries to predict randomly selected words from the paragraph or document. The trained embeddings were then applied to sentiment analysis classification and information retrieval tasks. Classification was performed on datasets such as the single paragraph categorization [37] and IMDB [38].

Recurrent neural networks (RNNs) and long short-term memory (LSTM) networks were used to account for sequence information and word dependencies in text classification. Vanilla RNNs suffer from the vanishing gradient problem, which LSTM addresses by introducing a memory cell. LSTM also features three gates: input, output, and forget gates, which regulate the flow of information into and out of the cells. While LSTM is effective at representing sequential information, it struggles with syntactic and semantic information. Tree-LSTM [39] was introduced to maintain syntactic information while performing classification. It was tested on the Stanford Sentiment Treebank dataset [40] for sentiment classification and outperformed various RNN, LSTM, and other models on fine-grained classification tasks, with comparable accuracy on binary classification.

Despite having memory cells, LSTM models often perform poorly on very long texts. Multi-timescale LSTM (MT-LSTM) [41] was introduced to capture long-term dependencies by grouping hidden states of LSTM with faster training. MT-LSTM outperformed other RNN, CNN, and LSTM models on the Stanford Sentiment Treebank [40], TREC [42], and IMDB datasets [43], with comparable results on binary classification tasks.

RNNs are effective for sequences where class depends on the entire sequence (global semantics), while CNNs are effective where class depends on local semantics. Dynamic convolutional neural network (DCNN) [44] takes text input embeddings as a matrix, performs wide convolution, and uses dynamic k-max pooling to convert variable-length input to a fixed-length feature map for further processing. Finally, a fully connected layer is used for classification. DCNN was tested on SST, TREC, and Twitter sentiment datasets and outperformed baseline models such as Naive Bayes, SVM, and NBOW.

A simple CNN [45] was used for text classification with various word-embedding approaches: random, static (word2vec), non-static (word2vec with training), and multi-channel (two word2vec vectors). The architecture includes a convolutional layer followed by max-over-time pooling and a fully connected head for classification with a softmax function. This model achieved better accuracy than other models on MR [46], SST-2, CR [47], and MPQA [48] datasets and comparable accuracy on SST-1, Subj [49], and TREC [42] datasets.

Character-level CNN [50] is a variant where the input is one-hot encoding of characters in the sequence. The input length is fixed at 1014 characters; sequences longer than 1014 characters are truncated, and shorter sequences are padded with zero vectors. Character-level CNN has 6 convolutional layers, pooling layers, and 3 fully connected layers with dropout layers for text classification. Models with different sizes of feature maps and hidden nodes were trained on eight datasets, outperforming previous machine learning models, LSTM, and word-level CNNs.

Other variations of CNNs are also developed like Multi-GPU-based CNN [51] for text classification, Gated CNN [52] architecture for text classification, a hybrid of CNN and LSTM called CNN-LSTM [53], double channel hybrid model called DC CNN-LSTM [54] to utilize word level and character level properties for text classification, and a very deep CNN called modified VDCNN [55] for Japanese text classification.

While CNNs perform well, they can lose important spatial information due to pooling operations, leading to misclassification. Capsule Neural Networks (CapsNets) [56] address this issue by using activity vectors to represent attributes of objects. The length of the activity vector indicates the presence of the object, and the direction specifies its features. CapsNets preserve information through layers, minimizing loss. CapsNets for text classification have been shown to outperform LSTM and CNN models on

datasets such as MR, Subj, CR, and AG, person text classification [57], and sentiment classification [58].

MCNN-LSTM [59] was introduced as a novel approach recently for text classification. MCNN-LSTM combined CNN and LSTM-based networks into a multi-class framework. This method addresses the challenges of document categorization, especially with imbalanced datasets that often hinder traditional classifiers. CNNs were responsible to extract features from text data while LSTMs handled the spatial structure of words within the documents. Authors applied the Tomek-Link algorithm to balance the dataset. The proposed model achieved an *F*1-score of 98% and an accuracy of 99.71%, outperforming existing methods.

The Context-Sensitive Multi-Tier Deep Learning Framework (CS-MDF) [60] represents an innovative approach to multimodal sentiment analysis that effectively processes contextual information across text, visual, and audio channels. This framework implements a distinctive three-tier architecture with specialized components at each level. The initial tier utilizes convolutional neural networks for textual analysis, three-dimensional CNNs for visual processing, and the openSMILE toolkit for extracting audio features. The second tier incorporates Bidirectional Gated Recurrent Units to identify and model relationships between utterances. The framework's final tier integrates the multimodal features and employs a BiGRU classifier for sentiment determination. Comprehensive evaluation across six diverse real-world datasets demonstrated CS-MDF's superior performance compared to ten contemporary models, with notable improvements in *F*1 scores, precision, accuracy, and recall metrics. The researchers further validated their approach through ablation studies and Gradient-weighted Class Activation Mapping visualizations, confirming both the framework's overall effectiveness and its successful alignment of features across different modalities.

RNNs, LSTMs, and GRUs are designed to capture long-term dependencies in text sequences but often fail with very long sequences. The attention mechanism [61] was introduced to address the long-term dependency problem by allowing the model to focus on important parts of the input sequence. Originally designed for sequence-to-sequence problems, the attention mechanism enables the encoder to capture input information in a context vector, and the decoder uses this vector to produce the output sequence. The attention mechanism was used in hierarchical attention networks [62] for grid classification. This model focuses on long sequences consisting of multiple sentences, with a word-based and a sequence-based encoder. The model, which uses bidirectional GRU encoders and attention scores, performed better than BOW, SVM, CNN, and LSTM-based models on Yelp reviews, IMDB reviews, Yahoo answers, and Amazon reviews datasets.

Recurrent networks, including LSTM and GRU, process input sequences word by word, making the next word dependent on the information from previous words. This sequential processing limits parallel computation and results in inefficient use of modern GPUs. To address this, the Transformer architecture [63] was introduced, enabling parallel processing by handling the complete sequence at once. It employs a positional embedding mechanism to maintain the order of words and introduces a self-attention mechanism that focuses on the relationships and dependencies among words in the input sequence. The Transformer model, designed as an encoder-decoder system, was notably successful in the WMT-14 machine translation task, surpassing previous state-of-the-art models in BLEU score.

The BERT architecture [64] represents a significant advancement in NLP, establishing unprecedented performance benchmarks across multiple language understanding tasks including Question Answering, NER, and text classification. This innovative model is available in two primary configurations: BERTBASE, featuring 110 million parameters structured across 12 transformer layers with a hidden representation size of 768 and 12 self-attention heads; and the more extensive BERTLARGE variant, which incorporates 24 transformer layers, expanded hidden representations of 1024 dimensions, and 16 self-attention heads for enhanced modeling capacity. BERT's training methodology follows a two-stage approach that has proven highly effective. The initial pretraining phase implements a masked language modeling objective on large volumes of unsupervised textual data, allowing the model to develop robust contextual representations. This is followed by a fine-tuning phase where the pre-trained parameters are adapted to optimize performance on specific downstream applications. This transfer learning paradigm has demonstrated remarkable efficiency in leveraging general language understanding for specialized tasks. Comparative analyses have consistently demonstrated BERT's superior performance relative to preceding language understanding frameworks such as Universal Language Model Fine-tuning (ULMFiT) [65]. BERT's bidirectional attention mechanism allows it to develop more comprehensive contextual representations by simultaneously considering both left and right contexts of each token, unlike earlier models that processed text directionally. This architectural advantage, combined with its transformer-based design facilitating parallel computation, has established BERT as a foundational model for contemporary NLP research and applications.

XLNet [66], another Transformer-based model, builds on BERT's architecture but is trained on a larger dataset of 113GB. This extensive training enables XLNet to outperform BERT across various tasks, though it requires significantly more training time and hardware resources. RoBERTa [67], an extension of BERT, was trained on an even larger dataset of 160GB and has demonstrated superior performance over both BERT and XLNet on several tasks.

These models can be fine-tuned for specific datasets. For instance, BERT, RoBERTa, and DistilBERT were fine-tuned for text classification on a legal domain dataset, where they outperformed attention-based LSTM and LSTM-based baseline models [68], with RoBERTa and DistilBERT achieving higher accuracy than BERT.

CNN-BiLSTM [69] introduced text classification model designed to address the increasing volume of smart tourism data. By combining CNN's feature extraction capabilities and BiLSTM's ability to capture long-term dependencies, CNN-BiLSTM was applied to sentiment analysis of tourist reviews. Compared with other models like BiLSTM and TextCNN, the CNN-BiLSTM achieved superior performance across multiple metrics, including precision (87.9%), recall (88.34%), and accuracy (85.1%). This study emphasizes the role of deep learning in enhancing information management in smart tourism by providing more accurate sentiment classification.

CI-AI [70] framework utilized transformer-based models for text classification in human-machine interaction tasks. By augmenting human-generated data with the T5 model through paraphrasing, the framework significantly expanded its training dataset. Seven transformer models, including BERT, RoBERTa, and XLNet, were fine-tuned and tested. It resulted in average classification accuracy improvement of 4.01% when trained on augmented data. The RoBERTa model delivered the highest performance, achieving 98.96% accuracy, which was further improved to 99.59% using an ensemble of transformer models

Short text classification remains challenging in NLP, with various models like convolutional and recurrent networks, and graph neural networks being applied. Authors [71] investigated the performance of different classifiers, including Transformers,

on benchmark datasets and two real-world datasets, NICE and STOPS. Their results showed that Transformers achieved state-of-the-art accuracy, questioning the need for specialized short text classifiers and positioning the NICE dataset as a valuable benchmark for future work.

PL-Transformer [72] represents a significant advancement in transformer architecture design, addressing fundamental limitations of traditional transformer models through its innovative dual-focused approach that enhances both linguistic awareness and representation utilization. Unlike conventional transformer implementations that primarily process token semantics in isolation from valuable linguistic structures, the PL-Transformer explicitly incorporates parts-of-speech information as complementary input, thereby enriching the model's understanding of grammatical relationships and syntactic patterns within textual data. Simultaneously, the architecture addresses the critical issue of information loss in standard transformers, which typically discard intermediate representations by utilizing only the final encoder layer for classification tasks, through its implementation of a sophisticated correlation coefficient attention mechanism (C-Encoder) that dynamically combines outputs from multiple encoder layers, preserving valuable information that would otherwise be lost in the encoding process. This comprehensive enhancement strategy has demonstrated measurable performance improvements across multiple benchmark datasets, with particularly notable results observed in the MPQA dataset evaluation, where the PL-Transformer achieved a substantial 3.95% accuracy improvement compared to baseline transformer implementations, thus validating the effectiveness of integrating linguistic knowledge and implementing layer-attentive mechanisms in transformer-based text classification systems.

R-Transformer_BiLSTM [73] was introduced to improve multi-label text classification, particularly for tasks with many labels and fine granularity. Traditional models struggle with sequence information and semantic depth as text length increases. The R-Transformer_BiLSTM model addressed these issues by using label embedding and attention mechanisms. It combined R-Transformer for capturing text information with part-of-speech embeddings, and employed BiLSTM+CRF for entity extraction. Keywords were identified through self-attention, and bidirectional attention and label embeddings refined the representations. The model demonstrated significant improvements in efficiency and accuracy on the RCV1-V2 and AAPD datasets.

A new SMS spam detection model [74] addressed the issue of spam, which poses risks like credential theft and data loss. This model used GPT-3 Transformer for advanced text embedding hence improving detection accuracy. An Ensemble Learning approach combined four machine learning models into one which enhanced the performance. Their model achieved 99.1% accuracy which was better than the previous counterparts.

A recent study [75] tried to make AI more accessible by evaluating prompt-based learning with transformer models for domain-specific classification. They compared zero-shot and few-shot methods with a fine-tuned model. The study used 2,088 human-labeled email responses where prompt-based learning achieved Cohen's kappa of 0.40, while fine-tuning reached 0.59. The study found that machine models provided reliability estimates, suggesting a hybrid approach where machines handle easy classifications and humans focus on difficult ones.

LongFormer [76] was introduced to handle long input context problem. Transformer-based models often have an input token limit of 512 or 1024 tokens per sample. LongFormer used 4096 input tokens to handle long sequence classification problems.

Other transformer-based models were also recently introduced, for example, T5 [77] for text classification and other text-related machine learning tasks, Big Bird [78] trained specifically for summarization and question answering, DeBERTa [79] with disentangled attention mechanism and enhanced mask decoder, Electra [80] with faster training by generator and discriminator approach upon tokens, and Ernie 2.0 [81] captures lexical, syntactic and semantic information during pretraining.

Transformer architecture is recently utilized in other than NLP domains. Swin transformer v2 [82] set benchmarks on image classification, semantic segmentation, object detection, and video action classification tasks. Vision Transformer (ViT) [83] provided the evidence that CNNs are not mandatory for computer vision and perform state-of-the-art performance on ImageNet, CIFAR-100, and VTAB dataset for image classification. Pyramid ViT [84] (PVT) combined CNN and Transformer to achieve better results than previous models on object detection, semantic, and instance segmentation tasks. Twins-PCPVT and Twins-SVT [85] took a simple-to-implement approach and used spatial attention for dense detection and segmentation. CrossViT [86] utilized dual-branch ViT that processes images at two different resolutions. By combining local fine-grained information from high-resolution patches with global context from low-resolution patches, CrossViT achieves superior performance in image classification. This cross-attention mechanism enables the model to capture a broader range of spatial relationships, making it effective for classification tasks on various datasets.

In our work, we fine-tune pre-trained BERT, RoBERTa, XLNet, and DistilBERT for message classification with additional classification layers at the end, producing baseline results on our dataset.

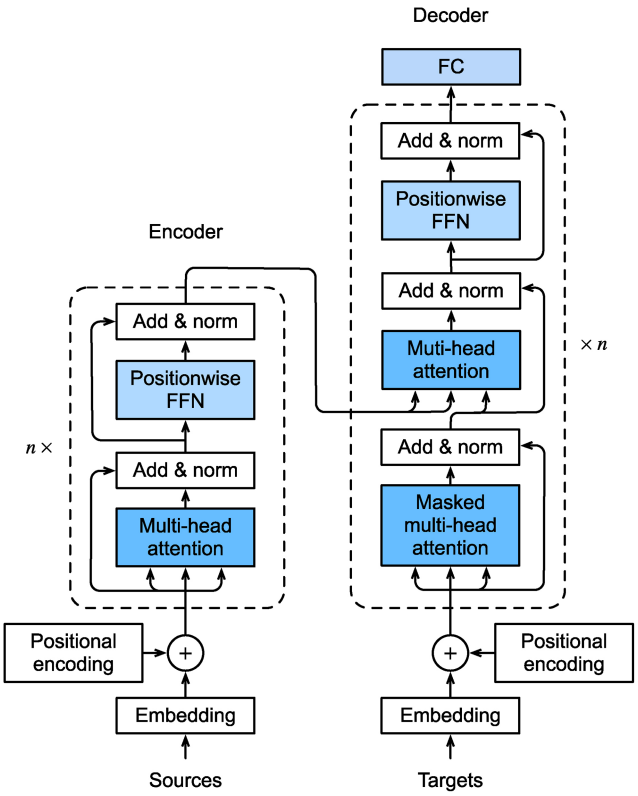
The attention-based cross-modal (ABCM) [87] transfer learning framework was introduced to enhance CVD prediction by integrating diverse data sources including clinical, imaging, and genetic information with an attention-driven mechanism. The ABCM framework uses a cross-modal attention system that selectively emphasizes the most relevant features from each data type, providing a more nuanced understanding of CVD. Extensive evaluations reveal that this model surpasses traditional single-source and multi-source methods in accuracy, precision, recall, and AUC, demonstrating its capacity to improve early and accurate CVD detection. The framework also shows resilience against missing data, further highlighting its robustness in handling complex multimodal datasets.

3. Proposed Methodology

Message classification is a multi-output classification problem where a text sequence x is assigned zero or more labels from a set of 5 classes $y = \{call - to - action, common - ground, establish - credibility, incentive - for - connecting, personalizati$. A message having all these five features is considered a high-quality message. In this work, we experiment with different machine learning and deep learning-based methods on the message classification task and produce baseline results on the message classification dataset. We have trained our transformer-based models, taking BERT, roBERTa, and XLNet as encoder part of our model. We then concatenate a classification layer at the end of the model to predict classes for messages.

Transformer-based models use transformer blocks as basic components. Transformer blocks use a self-attention mechanism to give attention to relevant parts of the sequence (see Figure 1 [88]). Transformer has an encoder and a decoder network. For classification tasks, we only use the encoder part of the network as part of our model.

Figure 1
Transformer architecture



The first layer of transformer-based models is the Input Embedding layer that takes the input IDs of the input text from the vocabulary of the tokenizer. The Input Embedding layer learns the embeddings for each token in the vocabulary during training. Unlike recurrent nets, the input text tokens are given in parallel to the Input Embedding layer, so there is no sequence information in the input. To keep the sequence information, positional information is injected into the input. The positional information is calculated by:

$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}) = \sin\left(\frac{\mathrm{pos}}{10000^{2\mathrm{i}/\mathrm{d}_{\mathrm{model}}}}\right) \tag{10}$$

$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}+1) = \cos\left(\frac{\mathrm{pos}}{10000^{2\mathrm{i}/\mathrm{d}_{\mathrm{model}}}}\right) \tag{11}$$

The positional vectors have the same dimensions as the token embeddings. The positional encodings are concatenated with the target token embeddings. This way, the positional information is injected with each token. These embeddings are referred to as positional input embeddings.

After encoding the positional information to each token, the input matrix X (text as a sequence of tokens with positional information) is converted to query, key, and value matrices. Each row in the query, key, and value matrices corresponds to the tokens in the sequence. Query, key, and value matrices have their own weight matrices that are learned during the training process by backpropagation (see Equations (12), (13) and (14)).

$$\mathrm{query} = \mathrm{X} * \mathrm{W}_{\mathrm{query}} \tag{12}$$

$$\mathrm{key} = \mathrm{X} * \mathrm{W}_{\mathrm{key}} \tag{13}$$

$$\mathrm{value} = \mathrm{X} * \mathrm{W}_{\mathrm{value}} \tag{14}$$

The attention score is calculated by Equation (15) from query, key, and value matrices:

$$\mathrm{Z}_{\mathrm{i}} = \mathrm{softmax}\left(\frac{\mathrm{Q}_{\mathrm{i}}\mathrm{K}_{\mathrm{i}}^{\mathrm{T}}}{\sqrt{\mathrm{d}_{\mathrm{k}}}}\right)\mathrm{V}_{\mathrm{i}} \tag{15}$$

The attention scores describe how much attention each token in the text sequence will give to every other token in the same text sequence. These scores are fundamental to the transformer architecture as they enable the model to focus on relevant parts of the input when generating each element of the output. The attention mechanism creates a weighted sum of all input tokens, with weights determined by their relevance to the current token being processed.

The query, key, value matrices, and attention score combined is called the self-attention mechanism. This mechanism allows each position in the sequence to attend to all positions in the previous layer, creating a rich representation that captures contextual relationships between words regardless of their distance from each other in the sequence. This is a significant improvement over RNNs, which struggle with long-range dependencies.

The self-attention mechanism can be converted to multi-head attention by using more than one query, key, and value matrix and their own corresponding weight matrices. Each attention head can focus on different aspects of the input, such as syntactic or semantic relationships, providing a more comprehensive representation of the input sequence. Multiple attention heads working in parallel allow the model to jointly attend to information from different representation subspaces at different positions.

The output of each head is concatenated to get the output vector, combining the diverse perspectives captured by each attention head into a unified representation. This concatenation allows the model to integrate information across different attention patterns, enriching the representation with multiple viewpoints on the same input sequence.

A linear layer is applied to the output vector, which helps project the concatenated outputs back to the original dimensionality and allows for further transformation of the representation. This projection layer serves as a way to combine the information from multiple heads and prepare it for subsequent processing steps.

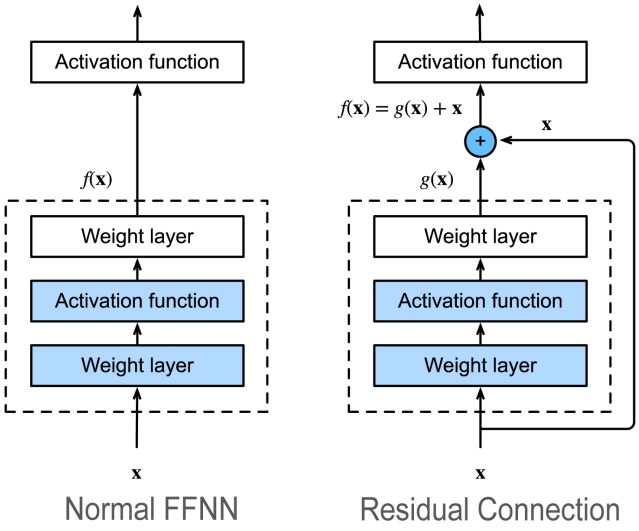
The linear layer combined with multiple self-attention heads is called multi-head attention. This complete mechanism forms the core of transformer models' ability to process sequential data without recurrence or convolution, relying instead on attention to establish relationships between tokens in the sequence.

The output vector of multi-head attention is point-wise added to positional input embeddings to form a Residual Connection. The Residual Connection helps in faster convergence and tackles vanishing gradient problems by skipping some layers and feeding the input directly to the later layers in the network (see Figure 2 [88]).

The output of the residual layer is passed through Layer Normalization. Layer Normalization takes the output/activations x_i having K dimensions of each individual training example and normalizes it to mean 0 and standard deviation 1 using the following equations to convert it to $\hat{\mathrm{x}}_{\mathrm{i}}$ vector with dimensions K:

$$\mu_{\mathrm{i}} = \frac{1}{\mathrm{K}}\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{x}_{\mathrm{i,k}} \tag{16}$$

Figure 2
Simple feed-forward network vs residual connection feed-forward network



$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(x_{i,k} - \mu_i)^2 \tag{17}$$

$$\widehat{x_{i,k}} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} \tag{18}$$

After normalization, the normalized activations are passed through a linear layer with learnable parameters γ and β (see Equation (19)):

$$y_i = \gamma\widehat{x_i} + \beta \equiv LN_{\gamma,\beta}(x_i) \tag{19}$$

Layer Normalization is used to stabilize the training process.

The output of the previous step is passed through a point-wise feed-forward neural network, which is basically a layer of two stacked linear layers applied to each position in the sequence. The output of the first linear layer is also passed through the ReLU activation function (see Equation (20)):

$$FFN(y_i) = \max(0, y_i W_1 + b_1)W_2 + b_2 \tag{20}$$

The point-wise feed-forward component is used to further process the information. The point-wise feed-forward layers also use the residual connection and layer normalization to stabilize the training process and help the model converge quickly.

All these components combined collectively make up the encoder of transformer-based models. Classification heads are attached to the output of this layer. In our case, 5 heads are attached for each category. The classification head consists of a dense layer. The complete architecture of the models is as follows (see Figure 3)

4. Experimental Results

In this section, we present the dataset, experimental setup, and results of our proposed approach.

4.1. Dataset

The message classification dataset contains 97,710 messages, sourced from QLU.ai with user's permission. The data are collected via LinkedIn through the QLU.ai product. Each message is labeled with one or more of five classes: call to action, common ground, establish credibility, incentive for connecting, and personalization. These labels capture specific elements within recruitment messages, enabling nuanced classification.

Call to Action: The message includes a phrase or sentence prompting further engagement, such as a meeting venue, a calendly/pencilit link, or phrases like "let's connect".

Common Ground: Indicates shared experiences, such as working at the same company, common educational backgrounds, hobbies, or interests.

Establish Credibility: Describes statements that reinforce the recruiter's credibility, the company, or the job role.

Incentive for Connecting: Highlights incentives or benefits associated with the role to encourage the prospect to connect.

Personalization: Reflects how tailored the message is, often including the recruiter's name, the prospect's name, or references to the prospect's current or previous company or job.

Figure 3
Complete architecture of the model

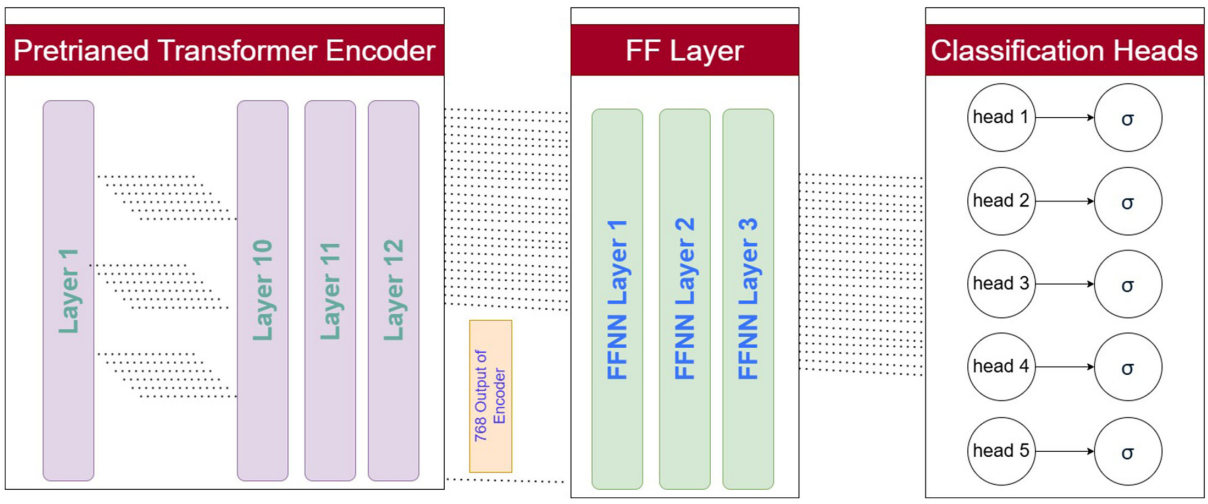


Table 1
Disk space

	Space	System occupied	Available
On CPU	108GB	39GB	70GB
On GPU	79GB	39GB	40GB

Table 2
Memory (RAM) information

	MemFree	MemAvailable
~13.30 GB	~10.61	~12.46

Table 3
CPU information

	Model name	CPU MHz	Cache size	CPU cores
CPU: 0	Intel(R) Xeon(R) CPU @ 2.20GHz	2200.162	56320 KB	1
CPU: 1	Intel(R) Xeon(R) CPU @ 2.20GHz	2200.162	56320 KB	1

The dataset was labeled based on 332 unique sentence fragments identified within the messages. Labeling was done by QLU.ai's team, who manually assigned labels to each fragment according to the relevant class definitions. If a message included fragments from multiple categories (e.g., Call to Action, Personalization, and Incentive), all applicable labels were assigned to that message. An example labeled message from the dataset is as follows:

> "This is (firstname) from (my company), the only data science solutions provider in the region. I think given your work experience at (their company name), you should be a part of this revolution. Are you connected to (my company)? Our company is growing its team each day, and we would like to add to it even more. How about if I discuss this role further with you?"

In this example:

Call to Action: "How about if I discuss this role further with you?"

Personalization: "(firstname)", "(my company)", and "I think given your work experience at (their company name)"

Establish Credibility: "The only data science solutions provider in the region"

Incentive for Connecting: "Our company is growing its team each day"

The dataset, after tokenization with the gensim tokenizer, contains 723 unique words. The class distribution was balanced initially but showed a small number of unique sentence pieces and vocabulary, resulting in a condensed data distribution. This led to near-perfect test accuracy, which raised concerns about generalization. To mitigate this, additional annotated data with diverse real-world examples was added to the test set, bringing it to 3,700 samples while leaving 94,010 samples for training

4.2. Experimental setup

We used Google Colab for all machine learning and deep learning experiments. Google Colab is a cloud-based platform provided by Google Research. At the time of this research, Google provides three different plans for colab: Colab free, Colab Pro, and Colab Pro+. We used the free version of colab for training our models. Even though the specification for the free version of colab changes with time and also depends on the availability of the GPU. The hard disk space Table 1, memory information Table 2, CPU information Table 3, and GPU information are as follows during this research (see Figure 4).

The GPU can be turned on or off in the Colab environment. The CPU and memory specifications are not affected by the state of the GPU; however, the disk space specifications change based on whether the GPU is on or off.

We performed multiple experiments with machine learning and deep learning algorithms, varying hyperparameters, and logged the best empirical accuracy scores.

4.3. Results

We trained three models. Three transformer-based models, i.e., BERT, RoBERTa, and XLNet, were used in the encoder block of our model and logged the accuracy in Table 4.

We trained the models for 1 epoch due to the simplicity of the problem and time constraints posed by colab. Since the message

Figure 4
GPU information

```
Wed Jun 29 09:31:29 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   47C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Table 4
Transformer-based model results

Model	Epochs	Input Size	Batch_size	Accuracy %	training time	GPU
RoBERTa based	1	128 tokens	8	94.84	3 h 49 m 05 s	K80
BERT based	1	256 tokens	8	95.67	4 h 34 m 57 s	K80
XLNet based	1	256 tokens	8	95.35	1 h 01 m 51 s	T4

Table 5
Machine learning-based algorithm results

	Input Size	Word_embeddings	Accuracy %
DecisionTree	300	GloVe	53.05
DecisionTree	100	GloVe	53.35
LinearSVM	100	GloVe	85.19
LinearSVM	**300**	**GloVe**	**90.92**
LogisticRegression	100	GloVe	83.97
LogisticRegression	300	GloVe	89.78
GaussianNB	100	GloVe	28.73
GaussianNB	300	GloVe	29.84
KNN Classifier	300	GloVe	76.59
RandomForest	100	GloVe	71.22

lengths are shorter, we kept the input token size at 128 or 256 and the batch size at 8. These networks have millions of parameters, and we only had 12GB of memory available for our model and data. Initially, we were assigned the K80 GPU by Colab for training RoBERTa and BERT models, which was slower than the T4 GPU. When training XLNet, Google Colab assigned a T4 GPU, making the training time of XLNet around 4 times shorter than the training time of BERT and RoBERTa. The accuracies of these models are close to each other. These models use attention mechanisms to better capture the context and semantic relationships between words. We achieved the highest accuracy of 95.67% with the BERT model, although the training time was slightly higher than that of RoBERTa. Both were trained on the same K80 GPU. Our best model, from an accuracy standpoint, is BERT-based model.

5. Comparative Study

We have trained different machine learning and deep learning-based algorithms to compare with our proposed approach.

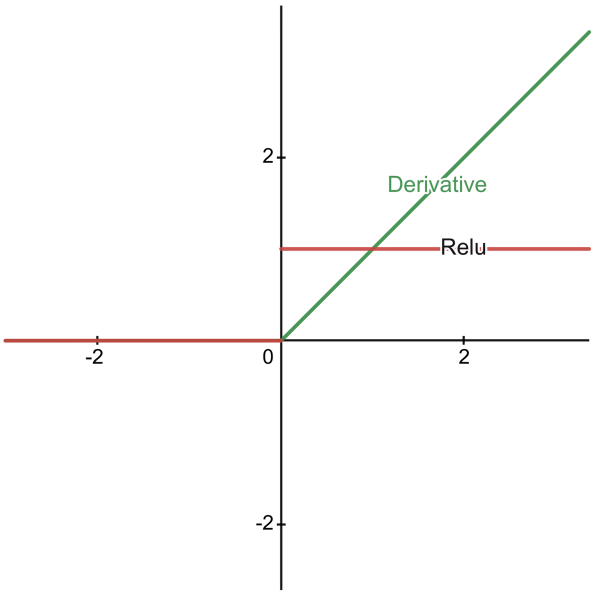
5.1. Machine learning algorithms

We have trained 6 famous machine learning algorithms on our dataset: DT, Linear SVM, LR, Gaussian Naive Bayes, KNN, and RF and logged the results in Table 5. We have used 300- and 100-dimension GloVe word embeddings. We calculate sentence embeddings by taking the average/mean of word vectors of the target sentence.

5.2. Deep learning algorithms

We train simple feed-forward neural networks with different hyperparameter configuration settings. We experiment with 100 and 300 dimensions of GloVe word embeddings. The sentence embeddings are calculated as the component-wise mean of word embeddings in the sentence. The network contains five heads: one for each classification category. We use the ReLU activation function (see Figure 5 and Equation (21)) on each layer as empirically it is proven to work well. ReLU is also the simplest and fastest to compute in forward and backward passes during backpropagation.

Figure 5
ReLU function and ReLU derivative



$$\mathrm{ReLU}(x) = \max(0, x) \tag{21}$$

The derivative function (see Figure 5 and Equation (22)) of ReLU is undefined at $x = 0$. When $x = 0$, the derivative is considered as 0.

$$\frac{\mathrm{d}}{\mathrm{dx}}\mathrm{ReLU(x)} = \begin{bmatrix} 0 & \mathrm{x} < 0 \\ 1 & \mathrm{x} > 0 \end{bmatrix} \tag{22}$$

Each classification head in the network is a binary classifier having a sigmoid function (see Figure 6 and Equation (23)) as the loss function. The cumulative loss is the sum of the losses at all classification heads.

$$\mathrm{Sigmoid(x)} = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}} \tag{23}$$

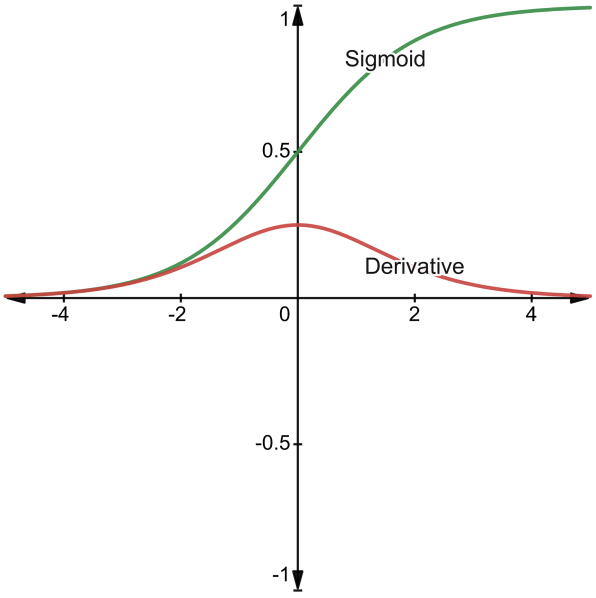
The derivative of the sigmoid function is calculated as:

$$\frac{\mathrm{d}}{\mathrm{dx}}\mathrm{Sigmoid(x)} = \mathrm{Sigmoid(x)}(1 - \mathrm{Sigmoid(x)}) \tag{24}$$

We used dropout with a probability of 0.5 on different layers of the network. Dropout helps in regularizing the network to avoid overfitting. Dropout simply drops neurons from the network with the specified probability (see Figure 7). We use the Adam optimizer to optimize the weights of the network. Adam is a combination of RMSProp and Momentum algorithms.

We compared our transformer-based models with different machine learning and deep learning techniques. We present the comparative study in this section.

Figure 6
Sigmoid function and Sigmoid derivative



The results of the LinearSVM are provided in the Table 4. LinearSVM with 300-dimensional embeddings achieved the best accuracy of 90.92%.

We performed multiple experiments with feed-forward networks with different hyperparameters but we also kept some hyperparameters constant in all experiments (see Table 6).

These hyperparameters are kept fixed as their counterparts works almost the same. Keeping these hyperparameters also reduce our search space. We used GloVe embeddings as it generally gives best results over other embedding algorithms like Word2vec. Generally, a neural network with large number of layers results in better performance but the data distribution space is smaller in our case. We keep the number of layers minimum. We heuristically select four hidden layers in our network. Empirically speaking, research shows that ReLU works best as activation function in hidden layers. Sigmoid loss function is better option when we have 2 number of classes. Adam optimizer is the gold standard of all optimization algorithms and learning rate is selected as 0.001 arbitrarily.

The results for feed-forward neural networks are shown in Table 7. Larger embedding sizes provide more information, which is why 300-dimensional embeddings perform better than 100-dimensional embeddings. Applying dropout on the final layers increased accuracy and prevented overfitting.

LSTM models with different hyperparameters were trained, and the accuracy was logged in Table 8. Some hyperparameters were kept constant (see Table 9) as they did not significantly affect the model's accuracy.

Different experiments were performed and found that 10 epochs are enough for our task. Messages in our dataset are trimmed to keep the input size to 100 tokens to tackle all input sizes. The number of neurons in hidden layers is selected as 512 since the problem is not that complex, so we do not need any extra neurons as it would increase the training time and risk of overfitting. ReLU is used as the activation function, as research shows that it empirically converges faster. We also used dropout with a probability of 0.5 to overcome the problem of overfitting. Adam optimizer is chosen as it helps the model to converge faster and overcome local minima problems up to a good extent. The learning rate is chosen as 0.001 arbitrarily. We used gradient clipping because LSTM models often face the exploding gradient problem due to their backpropagation through time nature during parameter optimization. If the gradient size on a neuron exceeds 5, that

Figure 7
Standard network vs dropout applied network

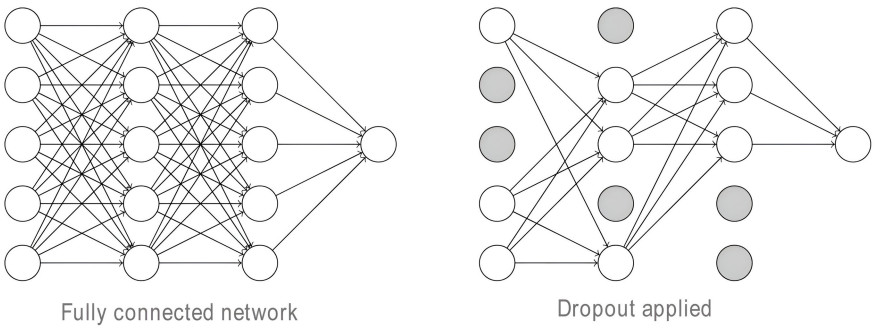


Table 6
Feed-forward neural network: Fixed hyperparameters

	n_layers	n_neuron	ActFunc	Batch size	Loss function	Optimizer	LR
Glove	4	[32, 64, 128, 256]	ReLU	1024	Sigmoid	Adam	0.001

Table 7
Feed-forward neural networks

	Input Size	Accuracy %	Dropout
10	100	79.73	No
10	300	85.73	No
30	100	72.22	on layer (1,2,3,4)
30	300	87.13	on layer (1,2,3,4)
30	100	89.03	on layer (2,3,4)
30	**300**	**92.76**	**on layer (2,3,4)**
30	100	89	on layer (3,4)
30	300	92.67	on layer (3,4)
30	100	90.54	on layer (4)
30	300	92.24	on layer (4)

Table 8
LSTM results

num LSTM layers	Bi-Direction	Batch_size	Accuracy %
1	No	1024	89.94
2	No	1024	93.97
3	No	1024	94.03
4	No	1024	93.95
1	Yes	1024	36.21
2	Yes	512	93.08
3	Yes	512	91.59
4	**Yes**	**512**	**94.14**

Table 9
LSTM fixed hyperparameters

Eps	Input Size	n_neuron	ActFunc	Loss Function	Dropout	Optimizer	LR	Gradient Clipping
10	100	512	ReLU	Sigmoid	Yes	Adam	0.001	Yes [31]

Table 10
Comparing our model with others

Model	Accuracy %
RoBERTa based	94.84
BERT based	95.67
XLNet based	95.35
LinearSVM	90.92
FFNN	92.76
LSTM	94.14

gradient value is clipped to 5; otherwise, it remains the same. We do not use any pre-trained word embeddings in LSTM. The model learns the embeddings during training itself.

We trained simple LSTM and bidirectional LSTM networks with different numbers of layers. We used a batch size of 1024 when the network was relatively smaller, but when the number of parameters increased due to the increasing number of layers or making the network bidirectional, we used a batch size of 512 due to the limited amount of memory in our experimental environment. Increasing the number of layers increased the model's capacity, thereby improving accuracy. We achieved our best accuracy score of 94.14% when the number of layers was 4 and the network was bidirectional. The bidirectional nature of the model allows it to consider the context from future words in the sentence as well. In contrast, the simple LSTM only looks at the context and relationships in the previous words in the sentence.

Finally, we present the comparison of all best models from different categories with our models in Table 10.

Our proposed model outperforms all other models, even when compared to their best-achieved results.

6. Conclusion

We worked on a multi-label classification problem in the domain of recruitment. We trained multiple machine learning and deep learning models on our dataset and found that the BERT model achieved better accuracy than all other models. The dataset consists of messages sent by recruiters to potential job candidates. Each message is labeled with 5 categories which define the strength of the message. Recruiters can use our trained models to assess the strength of the message before sending it to potential candidates. A strong message is more likely to get a response from prospects, thus increasing the chances of getting more candidates interviewed. Interviewing many candidates improves the chance of recruiting suitable employees for the organization.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by the author.

Conflicts of Interest

The author declares that he has no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Shaida Muhammad: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

[2] Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, *1*(4), 309–317. https://doi.org/10.1147/rd.14.0309

[3] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21. https://doi.org/10.1108/eb026526

[4] Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations.*

[5] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543.

[6] Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning, 32*(2), 1188–1196.

[7] Jothi Prakash, V., & Karthikeyan, N. K. (2021). Enhanced evolutionary feature selection and ensemble method for cardiovascular disease prediction. *Interdisciplinary Sciences: Computational Life Sciences*, *13*(3), 389–412. https://doi.org/10.1007/s12539-021-00430-x

[8] Vijay, A. A. S., & Prakash, J. (2022). A modified firefly deep ensemble for microarray data classification. *The Computer Journal*, *65*(12), 3265–3274. https://doi.org/10.1093/comjnl/bxac143

[9] Udhayakumar, S., Silviya Nancy, J., UmaNandhini, D., Ashwin, P., & Ganesh, R. (2021). Context aware text classification and recommendation model for toxic comments using logistic regression. In *Intelligence in Big Data Technologies—Beyond the Hype: Proceedings of ICBDCC 2019*, 209–217. https://doi.org/10.1007/978-981-15-5285-4_20

[10] Aliman, G., Nivera, T. F. S., Olazo, J. C. A., Ramos, D. J. P., Sanchez, C. D. B., Amado, T. M., …, & Valenzuela, I. C. (2022). Sentiment analysis using logistic regression. *Journal of Computational Innovations and Engineering Applications*, *7*(1), 35–40.

[11] Husnain, M., Khalid, A., & Shafi, N. (2021). A novel preprocessing technique for toxic comment classification. In *2021 International Conference on Artificial Intelligence,* 22–27. https://doi.org/10.1109/ICAI52203.2021.9445252

[12] Sharma, T., Diwakar, M., Singh, P., Lamba, S., Kumar, P., & Joshi, K. (2021). Emotion analysis for predicting the emotion labels using machine learning approaches. In *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering*, 1–6. https://doi.org/10.1109/UPCON52273.2021.9667562

[13] Hefny, A. H., Dafoulas, G. A., & Ismail, M. A. (2020). Intent classification for a management conversational assistant. In *2020 15th International Conference on Computer Engineering and Systems*, 1–6. https://doi.org/10.1109/ICCES51560.2020.9334685

[14] Ilyas, A., Obaid, S., & Bawany, N. Z. (2021). Multilevel classification of Pakistani news using machine learning. In *2021 22nd International Arab Conference on Information Technology*, 1–5. https://doi.org/10.1109/ACIT53391.2021.9677431

[15] Pearson, E. S. (1925). Bayes' theorem, examined in the light of experimental sampling. *Biometrika*, *17*(3/4), 388–442. https://doi.org/10.2307/2332088

[16] Zhang, H., & Li, D. (2007). Naïve Bayes text classifier. In *2007 IEEE International Conference on Granular Computing,* 708–708. https://doi.org/10.1109/GrC.2007.40

[17] Hairani, H., Anggrawan, A., Wathan, A. I., Abd Latif, K., Marzuki, K., & Zulfikri, M. (2021). The abstract of thesis classifier by using naive Bayes method. In *2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management,* 312–315. https://doi.org/10.1109/ICSECS52883.2021.00063

[18] Akanda, W., & Uddin, A. (2021). Multi-Label Bengali article classification using ML-KNN algorithm and Neural Network. In *2021 International Conference on Information and Communication Technology for Sustainable Development,* 466–471. https://doi.org/10.1109/ICICT4SD50815.2021.9396882

[19] Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. *Augmented Human Research*, *5*(1), 12. https://doi.org/10.1007/s41133-020-00032-0

[20] Al Sulaimani, S., & Starkey, A. (2021). Short text classification using contextual analysis. *IEEE Access*, *9,* 149619–149629. https://doi.org/10.1109/ACCESS.2021.3125768

[21] Vapnik, V., & Chervonenkis, A. Y. (1964). A class of algorithms for pattern recognition learning. *Avtomat. i Telemekh*, *25*(6), 937–945.

[22] Saigal, P., & Khanna, V. (2020). Multi-category news classification using support vector machine based classifiers. *Springer Nature Applied Sciences*, *2*(3), 458. https://doi.org/10.1007/s42452-020-2266-6

[23] Luo, X. (2021). Efficient English text classification using selected machine learning techniques. *Alexandria Engineering Journal*, *60*(3), 3401–3409. https://doi.org/10.1016/j.aej.2021.02.009

[24] Sudianto, S., Wahyuningtias, P., Utami, H. W., Raihan, U. A., Hanifah, H. N., & Adanson, Y. N. (2022). Comparison of random forest and support vector machine methods on twitter sentiment analysis (case study: Internet selebgram rachel vennya escape from quarantine). *Jurnal Teknik Informatika*, *3*(1), 141–145. https://doi.org/10.20884/1.jutif.2022.3.1.168

[25] Mukhtar, R., Iqbal, M. J., & Faheem, Z. B. (2021). Pakistani news classification based on headlines. *Pakistan Journal of Engineering and Technology*, *4*(4), 79–85. https://doi.org/10.51846/vol4iss4pp79-85

[26] Hussain, M. G., Hasan, M. R., Rahman, M., Protim, J., & Al Hasan, S. (2020). Detection of Bangla fake news using MNB and SVM classifier. In *2020 International Conference on Computing, Electronics & Communications Engineering,* 81–85. https://doi.org/10.1109/iCCECE49321.2020.9231167

[27] Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics*, 276–283.

[28] Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

[29] Syamala, M., & Nalini, N. J. (2020). A filter based improved decision tree sentiment classification model for real-time amazon product review data. *International Journal of Intelligent Engineering & Systems*, *13*(1), 191–202. http://doi.org/10.22266/ijies2020.0229.18

[30] Alzanin, S. M., Azmi, A. M., & Aboalsamh, H. A. (2022). Short text classification for Arabic social media tweets. *Journal of King Saud University-Computer and Information Sciences*, *34*(9), 6595–6604. https://doi.org/10.1016/j.jksuci.2022.03.020

[31] Qureshi, M. A., Asif, M., Khan, M. F., Kamal, A., & Shahid, B. (2023). Roman urdu sentiment analysis of songs' reviews. *VFAST Transactions on Software Engineering*, *11*(1), 101–108. https://doi.org/10.21015/vtse.v11i1.1399

[32] Kumar, D., Choubey, A., & Singh, P. (2020). Restaurant review classification and analysis. *Journal of Engineering Sciences*, *11*(8), 169–179. https://jespublication.com/login.php?id=1834

[33] Pericherla, S., & Ilavarasan, E. (2024). Transformer network-based word embeddings approach for autonomous cyberbullying detection. *International Journal of Intelligent Unmanned Systems*, *12*(1), 154–166. https://doi.org/10.1108/IJIUS-02-2021-0011

[34] Lim, S., Prade, H., & Richard, G. (2021). Classifying and completing word analogies by machine learning. *International Journal of Approximate Reasoning*, *132,* 1–25. https://doi.org/10.1016/j.ijar.2021.02.002

[35] Ning, G., & Bai, Y. (2021). Biomedical named entity recognition based on Glove-BLSTM-CRF model. *Journal of Computational Methods in Sciences and Engineering*, *21*(1), 125–133. http://doi.org/10.3233/JCM-204419

[36] Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé, III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 1*, 1681–1691.

[37] Haque, M. M., Biswas, N., Roy, N. S., Rafi, A. H., Islam, S. U., Lubaba, S. S., . . . , & Rahman, R. M. (2021). Data mining techniques to categorize single paragraph-formed self-narrated stories. In *ICT Analysis and Applications: Proceedings of ICT4SD 2020, 2,* 701–713. https://doi.org/10.1007/978-981-15-8354-4_70

[38] Li, G., Lin, Z., Wang, H., & Wei, X. (2020). A discriminative approach to sentiment classification. *Neural Processing Letters*, *51*, 749–758. https://doi.org/10.1007/s11063-019-10108-7

[39] Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, *1*, 1556–1566. https://doi.org/10.3115/v1/P15-1150

[40] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* 1631–1642.

[41] Liu, P., Qiu, X., Chen, X., Wu, S., & Huang, X. J. (2015). Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2326–2335. https://doi.org/10.18653/v1/D15-1280

[42] Li, X., & Roth, D. (2002). Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics.* https://doi.org/10.3115/1072228.1072378

[43] Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150.

[44] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 1*, 655–665. https://doi.org/10.3115/v1/P14-1062

[45] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1746–1751. https://doi.org/10.3115/v1/D14-1181

[46] Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* 115–124. https://doi.org/10.3115/1219840.1219855

[47] Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168–177. https://doi.org/10.1145/1014052.1014073

[48] Wiebe, J., Wilson, T., & Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, *39,* 165–210. https://doi.org/10.1007/s10579-005-7880-9

[49] Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* 271–278.

[50] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, *28*, 649–657.

[51] Ferjani, I., Hidri, M. S., & Frihida, A. (2022). Multi-GPU-based convolutional neural networks training for text classification. In *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference, 1,* 72–84. https://doi.org/10.1007/978-3-030-82193-7_5

[52] Sun, J., Jin, R., Ma, X., Park, J. Y., Sohn, K. A., & Chung, T. S. (2021). Gated convolutional neural networks for text classification. In *Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 2019,* 309–316. https://doi.org/10.1007/978-981-15-9343-7_43

[53] Mohd, N., Singhdev, H., & Upadhyay, D. (2021). Text classification using CNN and CNN-LSTM. *Webology*, *18*(4), 2440–2446. http://doi.org/10.29121/WEB/V18I4/149

[54] Liang, S., Zhu, B., Zhang, Y., Cheng, S., & Jin, J. (2020). A double channel CNN-LSTM model for text classification. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications,* 1316–1321. https://doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00169

[55] Moriya, S., & Shibata, C. (2018). Transfer learning method for very deep CNN for text classification and methods for its evaluation. In *2018 IEEE 42nd Annual Computer Software and Applications Conference, 2*, 153–158. https://doi.org/10.1109/COMPSAC.2018.10220

[56] Goldani, M. H., Momtazi, S., & Safabakhsh, R. (2021). Detecting fake news with capsule neural networks. *Applied Soft Computing*, *101,* 106991. https://doi.org/10.1016/j.asoc.2020.106991

[57] Kenarang, A., Farahani, M., & Manthouri, M. (2022). BiGRU attention capsule neural network for Persian text classification. *Journal of Ambient Intelligence and Humanized Computing*, *13*(8), 3923–3933. https://doi.org/10.1007/s12652-022-03742-y

[58] Chen, B., Xu, Z., Wang, X., Xu, L., & Zhang, W. (2020). Capsule network-based text sentiment classification. *IFAC-PapersOnLine*, *53*(5), 698–703. https://doi.org/10.1016/j.ifacol.2021.04.160

[59] Hasib, K. M., Azam, S., Karim, A., Al Marouf, A., Shamrat, F. J. M., Montaha, S., . . . , & Rokne, J. G. (2023). MCNN-LSTM: Combining CNN and LSTM to classify multi-class text in imbalanced news data. *IEEE Access*, *11,* 93048–93063. https://doi.org/10.1109/ACCESS.2023.3309697

[60] Paul, A., & Nayyar, A. (2024). A context-sensitive multi-tier deep learning framework for multimodal sentiment analysis. *Multimedia Tools and Applications*, *83*(18), 54249–54278. https://doi.org/10.1007/s11042-023-17601-1

[61] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations.*

[62] Zhang, Y., & Rao, Z. (2020). Hierarchical attention networks for grid text classification. In *2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence*, *1*, 491–494. https://doi.org/10.1109/ICIBA50161.2020.9277489

[63] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . , & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

[64] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1*, 4171–4186. http://doi.org/10.18653/v1/N19-1423

[65] Joseph, S., & Joshi, H. (2024). ULMFiT: Universal language model fine-tuning for text classification. *International Journal of Advanced Medical Sciences and Technology*, *4*(6), 1–9. https://doi.org/10.54105/ijamst.e3049.04061024

[66] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, *32*.

[67] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . , & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. In *International Conference on Learning Representations 2020 Conference.*

[68] Shaheen, Z., Wohlgenannt, G., & Filtz, E. (2020). Large scale legal text classification using transformer models. *arXiv Preprint:2010.12871*.

[69] Meng, L. (2024). The convolutional neural network text classification algorithm in the information management of smart tourism based on Internet of Things. *IEEE Access*, *12*, 3570–3580. https://doi.org/10.1109/ACCESS.2024.3349386

[70] Bird, J. J., Ekárt, A., & Faria, D. R. (2023). Chatbot Interaction with Artificial Intelligence: Human data augmentation with T5 and language transformer ensemble for text classification. *Journal of Ambient Intelligence and Humanized Computing*, *14*(4), 3129–3144. https://doi.org/10.1007/s12652-021-03439-8

[71] Karl, F., & Scherp, A. (2023). Transformers are short-text classifiers. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 103–122. https://doi.org/10.1007/978-3-031-40837-3_7

[72] Shi, Y., Zhang, X., & Yu, N. (2023). PL-Transformer: A POS-aware and layer ensemble transformer for text classification. *Neural Computing and Applications*, *35*(2), 1971–1982. https://doi.org/10.1007/s00521-022-07872-4

[73] Yan, Y., Liu, F. A., Zhuang, X., & Ju, J. (2023). An R-transformer BiLSTM model based on attention for multi-label text classification. *Neural Processing Letters*, *55*(2), 1293–1316. https://doi.org/10.1007/s11063-022-10938-y

[74] Ghourabi, A., & Alohaly, M. (2023). Enhancing spam message classification and detection using transformer-based embedding and ensemble learning. *Sensors*, *23*(8), 3861. https://doi.org/10.3390/s23083861

[75] Mayer, C. W., Ludwig, S., & Brandt, S. (2023). Prompt text classifications with transformer models! An exemplary introduction to prompt-based learning with large language models. *Journal of Research on Technology in Education*, *55*(1), 125–141. https://doi.org/10.1080/15391523.2022.2142872

[76] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv Preprint:2004.05150.*

[77] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., . . . , & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

[78] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., . . . , & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, *33*, 17283–17297.

[79] He, P., Liu, X., Gao, J., & Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations.*

[80] Clark, K., Luong, M., Le, Q. V., & Manning, C. D. (2019). ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations 2020 Conference.*

[81] Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., & Wang, H. (2020). Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI*

Conference on Artificial Intelligence, 34(5), 8968–8975. https://doi.org/10.1609/aaai.v34i05.6428

[82] Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., . . . , & Guo, B. (2022). Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 12009–12019.

[83] Alexey, D. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations 2021 Conference.*

[84] Wang, W., Xie, E., Li, X., Fan, D. P., Song, K., Liang, D., . . . , & Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 568–578.

[85] Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., . . . , & Shen, C. (2021). Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, *34*, 9355–9366.

[86] Chen, C. F. R., Fan, Q., & Panda, R. (2021). Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 357–366.

[87] Karthikeyan, N. K. (2024). A novel attention-based cross-modal transfer learning framework for predicting cardiovascular disease. *Computers in Biology and Medicine*, *170*, 107977. https://doi.org/10.1016/j.compbiomed.2024.107977

[88] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. UK: Cambridge University Press.

How to Cite: Muhammad, S. (2025). Enhancing Recruiter Outreach: Predicting Job Ad Quality with Advanced Machine Learning and NLP Models. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA52024083