Journal of Theoretical and Applied Information Technology
15th October 2024. Vol.102. No. 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# FROM CONTEXT-INDEPENDENT EMBEDDING TO TRANSFORMER: EXPLORING SENTIMENT CLASSIFICATION IN ONLINE REVIEWS WITH DEEP LEARNING APPROACHES

**KOMANG WAHYU TRISNA[1] , JINJIE HUANG [2*] , HESHAN LEI [3] , EDDY MUNTINA DHARMA [4]**

[123] School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China
[2] Key Laboratory of Advanced Manufacturing and Intelligence Technology, Harbin University of Science and Technology, Harbin, China
[14] Department of Informatic Engineering, Faculty of Information Technology and Desain, Primakara University, Bali, Indonesia
E-mail: [1]km.ayutrisna@gmail.com, [2]jjhuangps@163.com (corresponding author), [3]heshan961@outlook.com, [4]eddy@primakara.ac.id



| ID 55513 Submission | Editorial Screening | Conditional Acceptance | Final Revision Acceptance |
|---|---|---|---|
| 06-09-24 | 09-09-2024 | 23-09-2024 | 29-09-2024 |



**ABSTRACT**

The exponential progress in technology and the internet has resulted in an unparalleled surge in online engagement, where individuals openly express their viewpoints. Users provide a variety of opinions on politics, events, and product evaluations. User views wield substantial influence on decisions made by both companies and individuals. Manual procedures for identification become impracticable due to the large number of user opinions. Sentiment analysis techniques are employed as a resolution. Deep learning methods have demonstrated potential in accurately predicting polarity from internet reviews, outperforming standard models. Utilizing word embedding techniques in conjunction with deep learning models is crucial for attaining superior results in sentiment classification within the realm of natural language processing (NLP). Furthermore, word embedding approaches like Word2Vec and FastText are thoroughly analyzed for the purpose of mapping text to vectors composed of real numbers. In this study, every assessed deep learning model is combined with both context-independent word embedding and transformer-based embedding. The evaluation of the five model, each utilizing one of the five feature extraction approaches, is conducted using three datasets from distinct domains: IMDB, Amazon, and Yelp. The evaluation is based on multiple metrics, including accuracy, recall, precision, F1-score, and MCC.

**Keywords:** *Sentiment Analysis, Online Review, Deep Learning, Word Embedding, Transformers*

## 1. INTRODUCTION

The current development of social media is currently undergoing rapid and significant expansion. As a result, there is a significant increase in the continuous production of data as users share their opinions on various issues, such as items, themes, events, and breaking news [1], [2]. Social media has become a valuable tool for data analytics in various practical applications, especially when it comes to analyzing online evaluations. Businesses must now prioritize the examination of these comments to derive important insights related to their products or services. Individual viewpoints exert substantial impact on crucial decision-making processes. Over almost ten years, social media has become a well-established platform for individuals to openly share their thoughts and feelings. The expressions shared on social media can significantly influence the way specific items or businesses are perceived and managed online. Examining the sentiment patterns in consumer reviews can provide both a benchmark for other consumers and aid businesses in improving their service quality and overall customer satisfaction [3]. In the present day, individuals are no longer restricted to seeking advice from their acquaintances when they choose to purchase a consumer product. This is due to the abundance of user evaluations and debates available on public forums on the Internet. Businesses may no

longer need to engage in surveys, opinion polls, and focus groups to collect customer opinions, as there is an ample amount of publicly accessible information available. However, the process of locating and overseeing opinion websites on the Internet and extracting the information they provide continues to be challenging due to the widespread existence of many websites. Every website usually has a substantial amount of subjective text that might be challenging to understand due to lengthy blog posts and forum discussions. The normal human reader may struggle to find pertinent websites and effectively extract and condense the thoughts expressed within them. Therefore, there is a requirement for automated sentiment analysis systems.

Sentiment analysis largely focuses on the examination of opinions. It pertains to the automated analysis of subjective comments material to determine the emotional tone and propensity of the consumer [4]. Sentiment analysis is a prominent area of research in Natural Language Processing (NLP) that focuses on determining the polarity of textual data. It seeks to identify whether the sentiments expressed in a given text are positive, negative, or neutral [5]. Businesses utilize sentiment analysis to effectively understand social media comments, product evaluations, and diverse textual data sources. The capacity to discern sentiment is significant not only for individual decision-makers but also for business and governments alike. Being able to accurately perceive and understand the opinions and attitudes of the general public towards policies, products, and organizations can offer substantial benefits to these institutions, as well as to decision support systems and individuals[6]. During its first phases, sentiment analysis often depended on training shallow models with meticulously crafted features to attain good outcomes in polarity determination [7].

Text sentiment analysis is consist of three approaches: lexicon-based methods, machine learning-based methods, and deep learning-based methods. Lexicon-based methods assess the sentiment orientation of a text document by evaluating the meaning and sentiment conveyed by words and sentences using a predefined dictionary [8]. While effective, this approach poses a significant drawback in terms of the computational resources required to identify the sentiment orientation of each word in the dictionary. Conversely, traditional machine-learning methods often blend lexicon-based strategies with machine learning algorithms. These methods necessitate manual labeling, which can be time-consuming and resource-intensive. These labeled datasets serve as a training set for building classification models using popular supervised learning algorithms such as Naïve Bayes [9], Support Vector Machines (SVM) [10], k-nearest neighbour algorithm [11], latent Dirichlet allocation (LDA) [12], and random forest [13]. These methods often utilize linguistic features such as n-grams [14], part-of-speech (POS) tags, and lexical features. Machine learning methods offer a more automated and scalable approach to sentiment analysis in comparison to lexicon-based methods. While conventional machine learning models require human intervention to extract emotional features from input text, subsequent steps involve text vectorization and the utilization of traditional machine learning algorithms for sentiment classification [15]. However, coping with the exponential growth of data in social media repositories presents traditional algorithms with challenges when extracting sentiments from these vast datasets. Recent studies indicate that the utilization of deep neural networks in NLP tasks, such as sentiment analysis [16], language modeling [17], and machine translation, can substantially improve predictive performance.

Deep learning is a branch of machine learning that uses artificial neural networks with several layers to gain knowledge and generate predictions. Deep learning techniques have been extensively employed in many domains including image recognition [18], object detection [19], transportation [20], network optimization [21], sensor networks [22][23], system security [24], etc. Deep learning models are particularly well-suited for the analysis of text data sentiment due to their ability to effectively capture the intricate relationships between words and phrases. Deep learning, in contrast to conventional machine learning techniques, necessitates substantial data support rather than human-engineered features. Deep learning-based methods autonomously extract features from a variety of neural network models and enhance their performance by learning from their errors [25]. Word embeddings play a crucial role in the implementation of deep learning models. Word embeddings are commonly used as a fundamental input representation for deep learning models across a diverse array of NLT tasks. They have a vital role as a key component in deep models, providing input features for language-related tasks such as sequence labeling and text classification. In recent years, there has been a significant rise in the development of word embedding techniques, specifically classified as classic and context-based word embeddings. Choosing the appropriate approach for sentiment

analysis is essential and necessitates careful consideration. This study conducts experiments to thoroughly investigate the integration of context-independent embeddings and transformer-based embeddings within deep learning models, establishing a foundation for identifying the most suitable model for sentiment analysis development. Despite advancements in sentiment analysis techniques, traditional methods—such as lexicon-based approaches and conventional machine learning models—struggle to capture contextual information and manage the exponential growth of data in online reviews.

This study addresses a critical research gap: the need for a systematic comparison between traditional context-independent embeddings (e.g., Word2Vec and GloVe) and advanced transformer-based models (e.g., BERT and ALBERT) within deep learning frameworks. By conducting this comparison, we aim to provide a comprehensive understanding of the respective strengths and limitations of these embeddings in capturing sentiment nuances across diverse online review datasets.

The primary objective of this study is to explore and compare the effectiveness of context-independent and transformer-based embeddings in sentiment classification tasks utilizing deep learning models. Specifically, we seek to answer the following research questions:

1. How do context-independent embeddings perform compared to transformer-based embeddings in sentiment classification tasks?
2. What impact does the choice of word embeddings have on model performance across different datasets and domains?
3. What are the best practices for integrating these embeddings into deep learning models for optimal sentiment classification performance?

Additionally, this study makes the following contributions:

1. **Evaluation of Embeddings:** We systematically evaluate the performance of context-independent embeddings (e.g., Word2Vec, GloVe) and transformer-based embeddings (e.g., BERT, RoBERTa) using deep learning models for sentiment classification.
2. **Dataset Variability:** We utilize multiple datasets from diverse domains to test the generalizability and robustness of our models.
3. **Impact of Embedding Choices:** We analyze how different embeddings influence model outcomes, providing insights into their applicability and effectiveness in various contexts.

The selection of deep learning models (e.g., CNN, BiLSTM) is motivated by their demonstrated efficacy in capturing textual features. Furthermore, the incorporation of transformer-based models addresses the limitations of traditional embeddings in understanding context and semantics.

The subsequent sections of this work are structured in the following manner: Section II outlines the significant advancements made in the field of text sentiment analysis. Section III provides a comprehensive explanation of our proposed model. Section IV details the experimental procedure and the outcomes obtained from validating our model. Section V conclusion and future directions.

## 2. RELATED WORKS

Within the domain of Natural Language Processing (NLP), numerous investigations have leveraged neural language models and deep learning architectures. This section provides a concise overview of prior research endeavors focusing on sentiment analysis using deep learning methodologies.

### 2.1 Sentiment Analysis

Sentiment analysis is used to determine the overall attitude of a text or review, whether it is positive, negative, or neutral. This is done by analyzing the dominant emotional viewpoint expressed. In the early stages of sentiment analysis research, supervised machine-learning techniques were commonly used as the main classification or clustering modules [26]. Supervised machine-learning techniques used n-gram features and the bag-of-words (BOW) model to represent and categorize user-generated, sentiment-bearing texts [27]. These characteristics are suggested as solutions to tackle the limitations of the basic BOW model, such as disregarding word order and syntactic structures [28]. One major disadvantage of utilizing n-gram features is the significant increase in the dimensionality of the feature space. In recent studies, researchers have extensively explored feature selection methods to address this problem [29], [30]. Supervised methods like SVM, LDA, Naïve Bayes, and artificial neural networks are widely used for sentiment analysis and have shown impressive performance[31]–[33]. In supervised methods, sentiments of reviewers are predicted by analyzing the labeled sentiments of existing review data [34].

These methods have certain limitations as they necessitate a substantial amount of training data to encompass all potential scenarios and have a tendency to operate at a slower pace. In addition, supervised learning models are often tailored to specific domains. For instance, classifier models that are trained on laptop reviews may not perform optimally when applied to movie reviews [35]. To overcome these limitations, researchers have put forth unsupervised lexicon-based methods [36]–[38]. Lexicon-based methods utilize the sentiment orientation of words or phrases present in a review to assess the overall sentiment score. Hence, lexicon-based approaches depend on tallying sentiment lexicons instead of data training, and the effectiveness of the model can be enhanced by expanding the lexicon dictionary with a larger number of words. Several pre-existing dictionaries contain terms and their corresponding costs for sentiment analysis, including SentiWordNet [39], MPQA subjectivity lexicon [40] and LIWC lexicon [41]. However, lexicon-based methods are heavily dependent on the lexicon's quality and coverage, which can result in lower accuracy when compared to supervised approaches [42], [43]. In addition, the sentiment orientation of a word can differ depending on the domain, which can limit the effectiveness of lexicon-based methods in domains without specific lexicons. One possible solution to this problem is to analyze the emotional meaning of a word within its specific context [44]. Machine learning approaches can learn specific patterns from text, resulting in improved classification results. However, a limitation of these approaches is that they often rely on extensive training datasets to achieve optimal performance. Moreover, a classifier trained on a specific dataset may not achieve the same level of performance when applied to a different domain [45], [46]. The constraints can be surpassed by deep learning.

### 2.2 Deep Learning for Sentiment Analysis

In recent years, deep learning has been widely utilized in sentiment analysis, in addition to its success in numerous application fields. The latest advancements in sentiment analysis have primarily concentrated on acquiring word embeddings and investigating different deep-learning models for classification and clustering purposes. The topic of natural language processing is closely linked to the use of deep learning models and word embeddings. Word embeddings are essential for deep learning models as they offer a structured representation of text, enabling the models to learn and generate accurate predictions. These embeddings are designed to capture not just the literal meaning of words, but also their contextual associations and resemblances. Word embeddings allow deep learning models to represent words as dense vectors of real numbers. This representation enables the models to analyze words and capture their semantic and syntactic nuances, improving their capacity to comprehend intricate language patterns [47]. In their research, Yoon et.al., [48] introduced a multi-channel lexicon-based model that integrates convolutional neural networks (CNNs) with bidirectional LSTM (biLSTM) to perform sentiment categorization. The effectiveness of their model is dependent on the rules derived from the sentiment orientation of the lexicon in the given context, which is specific to the domain. CNNs, as advanced artificial neural networks, have the capability to discern intricate features across diverse data forms, encompassing images and text. They have mostly been utilized in computer vision applications, namely in tasks such as picture classification, object identification, and image segmentation.

Convolutional Neural Networks (CNNs) have recently been utilized for text-related tasks. CNNs are used as local feature extractors in sentiment analysis applications, especially when local patterns like n-grams have a substantial impact on long texts. For instance, Kalchbrenner et al. [49] introduced a dynamic CNN model called DCNN for analyzing sentiment at the phrase level. The DCNN employs Dynamic K-Max pooling to capture the relationships between words. Johnson and Zhang [50], employed the BOW model in the convolutional layer and introduced a novel model called Seq-CNN to retain word information. In a recent study, Rezaeinia et al. [51] CNN model that utilized enhanced word embeddings to analyze sentiment at the document level. The authors of the study improved the pre-trained Word2Vec [52] and GloVe [53] embeddings by incorporating lexical, positional, and syntactical information. These upgraded embeddings were then utilized in the analysis of three distinct datasets consisting of short texts. Additional variations of CNNs utilized for sentiment analysis applications encompass charCNN [54], CNN-multichannel [55], CNN-LSTM [56], Ada-CNN [57], and many more. Hyun et al. [58] proposed TCNN, employs the proximity correlation between the target word and neighboring words to acquire knowledge about the impact of the context on the target words. Although CNNs are highly effective at identifying patterns within localized temporal or spatial data, they frequently struggle to capture sequential correlations.

On the contrary, Recurrent Neural Networks (RNNs) are tailored for sequential modeling, yet they do not possess the capability to extract information concurrently. Sentiment analysis of text can be viewed as a challenge of sequential modeling, in which RNNs are frequently effective in sequentially assessing lengthy texts. RNNs are commonly employed in text sentiment analysis due to their inherent properties. However, conventional RNNs have challenges related to the amplification and attenuation of gradients while handling lengthy data sequences. Long Short-Term Memory networks (LSTMs) [59] were devised as a variant of deep neural network architecture leveraging RNNs to address the vanishing gradient dilemma in particular. Conventional RNNs are unable to effectively process input sequences of any length. In order to overcome this constraint, LSTM models include forget gates. LSTMs and their variations are often used in sentiment analysis applications because to their capacity to capture long-term dependencies [60]. For example, Xu et al. [61] introduced the cached LSTM model to capture both local and global semantic information in lengthy text. Moraes et al. [62] proposed the utilization of p-LSTM, a model that incorporates three-word embeddings instead of single-word embeddings, in addition to a phrase embedding layer. The p-LSTM model utilizes LSTM for sentiment classification tasks.

In recent study, Gupta et al. [63], proposed an innovative multichannel LSTM model named SS-BED for emotion detection within Twitter data. Their approach involves the concurrent utilization of GloVe [53] and Sentiment Specific Word Embedding (SSWE) [64] in parallel as pre-trained word embeddings. Subsequently, three LSTM modules are successively employed for each pathway to capture extended dependencies within the text. In the final step, the output consists of two concatenated feature vectors, which serve as the input to the fully connected layer. Additional LSTM variants employed in sentiment analysis include TD-LSTM [65], SLSTM [66], cBLSTM [67], Tree-LSTM [68], and Sentic LSTM [69]. Ma et al. [69] introduced a modified version of LSTM dubbed Sentic LSTM to incorporate both dominant and recessive aspects in phrases. This model unit includes a unique output gate that is specifically designed to combine token-level memory and concept-level input. Chen et al. [70] developed an LSTM model for complex emotional interpretation of Chinese product reviews, using the mathematical principles of regression neural network. Wen et al. [71] introduced a hardware architecture for LSTM network employing memristor crossbars in a distinct study. Hu et al. [72] performed sentiment analysis on brief texts by creating a keyword vocabulary and incorporating it into the LSTM model.

Furthermore, bidirectional long short term memory (BiLSTM) [73] is an advancement of LSTM. The BiLSTM efficiently combines the hidden layers from both the forward and backward directions, allowing access to contextual information from both previous and subsequent elements. Therefore, BiLSTM outperforms LSTM in dealing with sequential modeling jobs. BiLSTM has been effectively utilized in recent applications for sentiment analysis tasks, resulting in significant accomplishments [74][75]. Another variant of RNN, the Gated Recurrent Unit (GRU), is frequently utilized for sentiment analysis tasks. GRU units possess a more streamlined structure, consisting solely of two gates: the reset gate and the update gate. The reset gate is responsible for selecting the information that needs to be discarded from the previous state, while the update gate regulates the information that should be retained from the current input and the prior state. GRU units do not have a separate memory cell like LSTM. Instead, they maintain a hidden state that combines both the previous memory and the current input information. GRU has fewer parameters compared to LSTM because it lacks the complexity of separate memory cells and additional gates. GRU's implementation can enhance computational efficiency and expedite training, particularly noticeable with smaller datasets. Due its simpler architecture, GRU often converge faster during training compared to LSTMs. This makes GRUs a popular choice when computational resources are limited or when training time is a critical factor. Yang et al. [76], introduced a new attention-based network called hierarchical attention networks (HAN) for text classification. Within their paradigm, they utilize two attention modules, one at the word level and another at the phrase level. The attention modules are arranged in a stack on top of the outputs of sequence encoders based on GRU.

Zhang et al. [77] introduced a model that utilizes bidirectional recurrent neural networks to handle multiple inputs and outputs. They employed two distinct biGRU layers in their model to provide part-of-speech and sentence representations. They subsequently merged lexical information by utilizing attention on the output of the softmax activation for the part-of-speech representation. Additionally, they enhanced the model's performance by integrating the probability of auxiliary labels as a feature, enabling the capture of essential correlations between the output labels and the hidden layers. In Li et al. [78]

a sentiment classification of restaurant reviews was proposed using an attention-based Bi-GRU neural network. The proposed model combines Word2Vec, Bi-GRU, and attention mechanisms to achieve better results with fewer parameters in building a sentiment analysis model. The model ensured the symmetry of the hidden layer weight update by utilising the important benefit of Bi-GRU. Pan et al. [79] introduced the concept of sentiment analysis in Chinese. The text sentiment was analyzed using a Bi-GRU and attention mechanism model. The suggested model can extract profound characteristics of the text and integrate the sentence's context to acquire text attributes with more precision. In addition, they introduced multi-head self-attention, a method that decreases dependence on external parameters. This method involves assigning weights to word vectors, emphasizing text characteristics, and giving greater consideration to the internal relationships within sentences.

Due to its lightweight nature, many researchers combine several deep learning methods with the GRU model to achieve better results. Luo [80] combined Latent Dirichlet Allocation (LDA) text representation with CNN. The LDA topic model trained the latent semantic space representation of the short text, producing a feature vector representation based on the topic distribution. Then, CNN combined with GRU is used as a classifier. Meanwhile, Ni et al. [81] integrated two distinct recurrent neural network (RNN) models, namely LSTM and GRU, with GloVe word embedding. The suggested model was implemented on the IMDB dataset. In addition, a hybrid approach utilizing both LSTM and GRU is employed for sentiment analysis in the field of cryptocurrencies [82]. Cheng et al. [83], suggested a sophisticated hybrid model that incorporates a multi-channel convolutional and bidirectional GRU with multi-head attention capsules. This model uses vector neurons instead of scalar neurons to represent text emotions and leverages capsules to describe text emotions. Multi-head attention is capable of encoding interdependencies among words, identifying sentiment terms in text, and employing CNN and Bidirectional Gated Recurrent Unit Network (Bi-GRU) to extract local characteristics and global semantic aspects of the text, respectively. This study aims to assess and identify the most appropriate deep learning models, specifically CNN, LSTM, BiLSTM, GRU, and BiGRU, for sentiment analysis.

### 2.3 Word Embedding

Deep learning models rely heavily on word embeddings as a source of input information for language tasks like text classification and sequence labeling. For this objective, a significant number of word embedding techniques have been presented in the last ten years, mostly divided into context-dependent and context-independent embeddings [84]. Word embedding is a language modeling technique used to associate words with numerical vectors, effectively representing words or phrases in multi-dimensional vector spaces. These word embeddings can be created through diverse techniques such as neural networks, co-occurrence matrices, probabilistic models, and more. Context-independent and context-dependent word embeddings are the two primary categories of word embeddings that have been created. Context-independent techniques, sometimes referred to as "classic" word embeddings, use co-occurrence matrix factorization or shallow neural networks to learn representation [85]. The learned representations are typified by being distinct and unique for every word, regardless of the word's context. For this reason, these embeddings are usually provided as downloaded files and pre-trained on generic text corpora. They can be used straight away to set the embedding weights for language tasks that come after. Among the notable examples are word2vec [52], GloVe [53] and FastText [86]. Context-dependent approaches, in contrast to context-independent ones, learn distinct embeddings for the same word based on the context in which it is used. For example, a term with several meanings like "bank" could have different embeddings depending on whether it's used about a river or something financial. There are two primary groups into which context-dependent word embedding learning techniques have mostly developed. Approaches based on RNNs, such as CoVe [87], Flair [88] and ELMo [89] make up one group.

On the other hand, it has been shown that recently created Transformer-based models [90], including Bert [91] and Albert [92], may effectively learn contextualized word representations. A word can be transformed into a vector that condenses its syntactic and semantic components using word embeddings. Thus, word embeddings are considered optimal feature representations for neural network models in later NLP applications, including machine translation including machine translation [87], sequence learning [88], text classification [93]–[96], etc. Word embeddings are frequently used in sentiment analysis to transform words into low-dimensional vectors that computers can process. Transformer-based embedding and context-independent word embedding will be the main topics of this study. We will assess these word embeddings

and see how deep learning models are applied to them.

## 3. METHODS

This section introduces context-independent embedding and transformer-based embedding models, along with their configurations. It also discusses the various deep-learning configurations employed in this work. The comprehensive evaluation model is depicted in Figure 1, while Figure 2 provides in-depth details regarding the model. The research methodology involves the following steps:

1. **Data Collection and Preprocessing:** We collected datasets IMDB, Amazon, and Yelp dataset, including data cleaning, tokenization, and text normalization.

2. **Model Selection and Training:** We implemented multiple deep learning architectures (CNN, BiLSTM, BiGRU) with context-independent and transformer-based embeddings. Hyperparameters such as learning rate, batch size, and dropout rate were optimized using a grid search.

3. **Evaluation Metrics:** Models were evaluated using standard metrics such as accuracy, precision, recall, and F1-score.

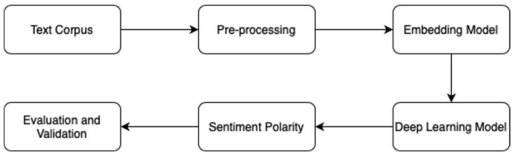


*Figure 1. Proposed Model*

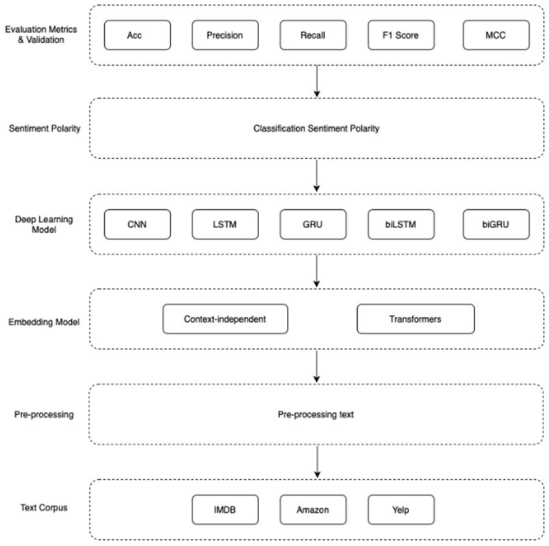


*Figure 2. Proposed Model in Details*

### 3.1 Text Preprocessing

Data preparation is a crucial aspect of natural language processing projects, as algorithms lack an inherent understanding of text. We must transform the language into numerical representations that algorithms can understand. Online evaluations frequently consist of textual content that can include extraneous data such as special characters, symbols, hyperlinks, and so on. Regular expressions are used to eliminate the disruptive noise in the information. As part of the preprocessing stage, all the reviews are divided into individual words, a process known as tokenization. Subsequently, redundant words are removed to create a distinct representation for each word. In this study, we convert all the sentences into lowercase to ensure that words are treated consistently, regardless of their original case. Since each word will be processed individually, it is necessary to carry out a separation process for each word. In addition, stopword removal is used to reduce noise by removing words that do not carry significant meaning, allowing the model to focus on words that are more informative and contextually relevant. After stopword removal, we join the remaining words back into a single text string with spaces separating them. The overall text preprocessing process is illustrated in Figure 3. Examples of a text review before and after the preprocessing process can be seen in Table 1.

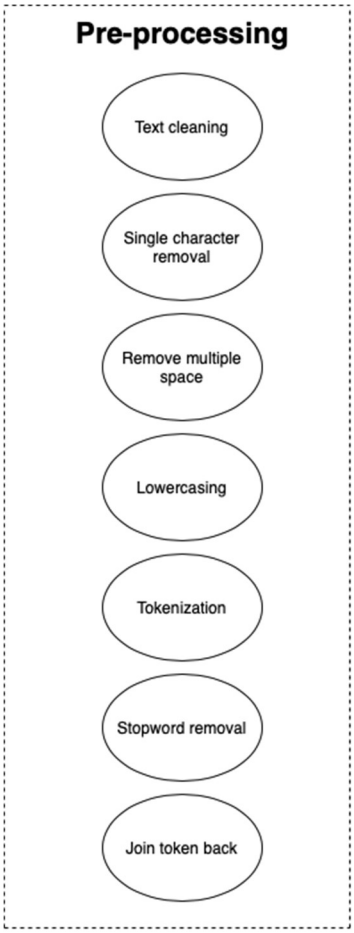


*Figure 3. Pre-processing framework*

*Table 1. Example of pre-processing Process on Text Review*

| Class | Review | After pre-processing |
|---|---|---|
| Positive | Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers us a vivid portrait about human relations. This is a movie that seems to be telling us what money, power and success do to people in the different situations we encounter. | petter mattei love time money visually stunning film watch mr mattei offers us vivid portrait human relations movie seems telling us money power success people different situations encounter |
| Negative | Bad plot, bad dialogue, bad acting, idiotic directing, the annoying porn groove soundtrack that ran continually over the overacted script, and a crappy copy of the VHS cannot be redeemed by consuming liquor. | bad plot bad dialogue bad acting idiotic directing annoying porn groove soundtrack ran continually overacted script crappy copy vhs cannot redeemed consuming liquor |

### 3.2 Embedding Layer

Word embeddings are a method of representing words in a vector space by grouping words that have similar meanings. Every word is allocated to a vector and subsequently acquired by a process resembling that of neural networks. The system acquires knowledge and selects a vector from a pre-established set of words. The size of the words can be determined by specifying it as a hyperparameter. This section will explain the embedding models that will be utilized in the assessment process, specifically fastText, word2vec, gloVe, bert, and Albert.

#### 3.2.1 fastText

FastText is an open-source library developed by Facebook's AI Research (FAIR) team. It is specifically built to handle the representation and categorization of textual data. It is tailored to efficiently handle large volumes of text data, providing features for tasks such as text classification, word representation, and computation of text similarity. FastText utilizes word embeddings, which involve dense vector representations of words within a continuous vector space. These embeddings excel at capturing both semantic and syntactic relationships among words, leveraging the distributional characteristics of words within a given corpus. FastText is a viable and efficient method for creating word embeddings from text documents. In this method, each word is represented by breaking it into several character n-grams [97]. FastText considers the inherent semantics of words, making it a suitable word embedding approach for languages with complex morphology and infrequent words. Its primary goal is to incorporate the internal structure of words rather than solely learning word representations. The uses of fastText can be seen in Figure 4.

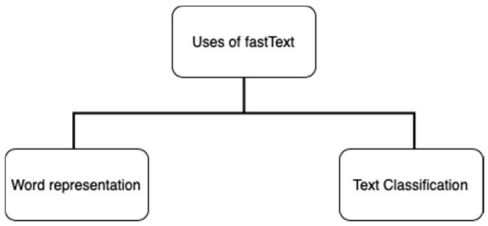


*Figure 4. The Uses of fastText*

FastText utilizes a sliding window approach to process the input text. This is accomplished through the process of learning either the main word based

on the context around it, which is called Continuous Bag of Words (CBOW), or by learning all the surrounding words based on a central word, which is known as Skipgram. The learning process can be understood as a sequence of modifications to a neural network composed of two layers of weights and three layers of neurons. In this configuration, the two outside layers are associated with a neuron for each word in the lexicon, while the middle layer consists of neurons that match the dimensions in the embedding space. FastText can learn vectors for character n-grams, which are sub-components of words. This is a feature that word2vec does not have. This guarantees that words such as 'love,' 'loved,' and 'beloved' will possess comparable vector representations, regardless of their occurrence in distinct contexts. This feature significantly improves the learning process, especially for languages that have a wide range of inflectional variants [98] .

The FastText model utilized in this study employs 300-dimensional word vectors. This implies that each word in the vocabulary is depicted as a vector in a 300-dimensional space. These word vectors encapsulate semantic details about words and their associations with other words in a continuous vector space, rendering them valuable for diverse natural language processing tasks. The fastText embedding layer in our model takes sequences of word indices as input. This layer generates a word embedding matrix of shape (vocabulary_size + 1, embedding_dim), where an additional row is added for out-of-vocabulary words. For each word index in the input sentence, the fastText embedding layer looks up the corresponding dense vector from the embedding matrix. This dense vector represents the embedding for that word. The output of the embedding layer for a sequence is a 3D tensor of shape (batch_size, input_length, output_dim), where each word in the input sequence is mapped to its corresponding dense vector in the output. These dense vectors serve as input to the next layer. We set the `trainable` parameter to 'false', indicating that the pre-trained fastText embeddings will not be updated during training. For further details, refer to Table 2, which outlines the fastText parameters used in this study.

*Table 2. fastText Parameters Used in The Proposed Model*

| Parameter | Value |
|---|---|
| input_dim | IMDB: 90704<br>Amazon: 59914<br>Yelp: 214392 |
| output_dim | 300 |
| weights | IDMB: (90704,300)<br>Amazon: (59914,300)<br>Yelp: (214392,300) |
| input_length | 300 |
| trainable | false |

### 3.2.2 Word2Vec

Word2Vec is a neural network architecture consisting of two layers, commonly used for token vectorization. Developed by Mikolov et al. [99], Word2Vec is a widely adopted technique for learning word embeddings from text data. It operates as an unsupervised method, efficiently extracting semantic relationships between words by analyzing their co-occurrence patterns within a given text corpus. This architecture comprises two models: the CBOW model and the skip-gram model. Together, they form a class of models within the Word2Vec framework [100]. In the CBOW model, the target word is determined using the context of each word as input, whereas in the skip-gram model, the context word is predicted based on the target word [101]. The skip-gram model demonstrates robust performance even with a small amount of data and can yield promising results in representing rare words [101]. On the other hand, the CBOW model is known for its swiftness and proficiency in representing commonly occurring words. The architecture of CBOW and skip-gram model can be seen in Figure 5.

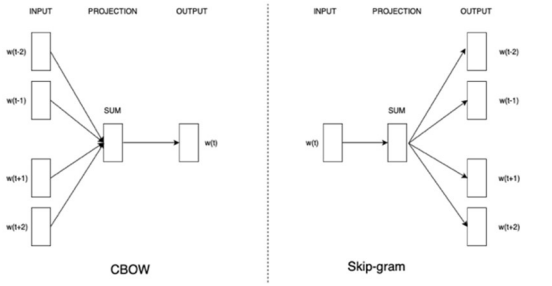


*Figure 5. Word2Vec CBOW and skip-gram architecture* [99]

Suppose we have a series of training words $w_1, w_2, \ldots, w_T$ with length of sequence is $T$, the skip-gram model's objective is determined by equation (1) [99]:

$$\text{argmax}_\theta \frac{1}{T}\sum_{t=1}^{T}\sum_{-C\le j\le C, j\neq 0} \log P_\theta(w_{t+j}/w_t) \quad (1)$$

Where $C$ represent the size of training context, $P(w_{t+j}/w_t)$ represent a neural network with a set of parameters denoted by $\theta$. In this model we used pre-trained word2vec model provided by Google News dataset. This model is typically pre-trained on a large corpus of Google News articles and is known for its

high-quality word embeddings. The word2vec model that we use here is 300-dimensional word embeddings. The fasttext parameters used in this study can be seen in Table 3. Unlike the parameters in fastText, here we make the values for input_dim and weight in each dataset the same.

*Table 3. Word2vec parameters used in the model*

| Parameter | Value |
|---|---|
| input_dim | 10000 |
| output_dim | 300 |
| weights | (10000,300) |
| input_length | 300 |
| trainable | false |

### 3.2.3 Glove

GloVe, also known as Global Vectors, is a word representation technique that has been created using an unsupervised learning method. Its purpose is to generate word embeddings by analyzing the co-occurrence matrix of words in a given corpus. GloVe can be regarded as an expansion of Word2Vec, specifically created to proficiently acquire word embeddings from textual sources. The model integrates the Word2Vec model's learning based on local context with global matrix factorization. When calculating the error function in the model, the probability ratios of words are also included. Words that exhibit a strong correlation in the text document and are prone to co-occur are deemed more significant than other words in the process of acquiring knowledge. GloVe is a tool that transforms text into word vectors, which are able to measure the distance between words [16]. The model can be characterized as a global log-bilinear regression model, and its goal function is represented by Equation (2) [53]:

$$J = \sum_{i,j}^{V} \int (X_{ij})(w_i^T \omega_j + b_i + b_j - \log X_{ij}) \qquad (2)$$

Where $V$ denotes the vocabulary size, $w \in R^d$ represent word vectors, $X$ denote co-occurrence matrix, and $X_{ij}$ denotes the number of times word $j$ occurs in the context of word $i$. $\int(X_{ij})$ denotes a weighting function and $b_i$, $b_j$ are bias parameters [53]. In this study we use 100-dimensional space with the 6 billion tokens. Details parameter of Glove embedding can be seen in Table 4.

*Table 4. Glove parameters used in the model*

| Parameter | Value |
|---|---|
| input_dim | IMDB: 112281<br>Amazon: 59914<br>Yelp: 214392 |
| output_dim | 100 |
| weights | IMDB: (112281, 100)<br>Amazon: (59914, 100)<br>Yelp: (214392, 100) |
| input_length | 300 |
| trainable | false |

### 3.2.4 Bert

In 2019, Devlin et al. [91] employed the transformer architecture to create an innovative language representation model called BERT (Bidirectional Encoder Representations from Transformers). This model ushered in a new era in NLP, demonstrating exceptional performance on a wide range of NLP tasks. Bert utilizes the unsupervised learning method to pre-train deep bidirectional representations from extensive unlabeled text collections, employing two novel pre-training objectives: masked language model (MLM) and next sentence prediction (NSP). BERT addresses the constraint of prior language models that only include one-way representations of words in a phrase. The model constructs a bidirectional masked language model that predicts words randomly masked in the phrase, hence enhancing the contextual information of the words. BERT comes in two variants: BERT-base, comprising 12 encoder layers, a hidden size of 768, 12 multi-head attention heads, and 110 million parameters; and BERT-large, featuring 24 encoder layers, a hidden size of 1024, 16 multi-head attention heads, and 340 million parameters. Both of these models have undergone training using data from English Wikipedia and BookCorpus. This study employed Bert-base to analyze deep learning models utilizing transformer-based approaches.

### 3.2.5 Albert

Albert, which stands for “A Lite Bert”, was made available in open source version by Google in 2020, developed by Lan et al [92]. Albert is a groundbreaking advancement in NLP architecture. By implementing innovative techniques such as parameter sharing across layers and factorized embedding parameterization, Albert drastically reduce the number of parameters compared to traditional transformer models while maintaining competitive performance. This reduction in parameters leads to improve memory efficiency and faster training times, making Albert ideal large-scale NLP tasks. Despite its compact size, Albert demonstrates remarkable effectiveness in various language understanding tasks, including language translation, text summarization, sentiment analysis, and question answering. Its efficiency and scalability mark a significant step forward in NLP research, promising more resource-efficient and scalable models for real-word applications.

### 3.3 Deep Learning Model

#### 3.3.1 Convolutional Neural Network (CNN)

CNNs are deep learning models specifically created to identify intricate patterns and characteristics inside datasets. They excel in extracting features from image and text data. CNNs have been widely used in computer vision applications, particularly in areas such as picture classification, object identification, and image segmentation. CNNs are deep learning algorithms primarily designed for processing images and videos. They take images as inputs, extract and learn features, and classify them based on these learned features. Recently, CNNs have also been adapted for text analysis, demonstrating versatility across different types of data. The CNN model typically operates in two steps: feature extraction and classification. Throughout the feature extraction process, a multitude of filters and layers are employed on input images to glean detailed information and features. Once this phase is complete, the extracted features are passed to the classification stage, where the images are categorized based on the target variable of the problem.

A standard CNN architecture consists of an input layer, an output layer, and hidden layers. The hidden layers typically comprise convolutional layers, where feature maps are generated from the input data through convolution operations, along with pooling layers and fully connected layers. Activation functions, such as ReLU, are used in conjunction with these feature maps to introduce non-linearity into the architecture. This enhances the network's ability to model complex relationships in the data. In the pooling layers, the outputs from clusters of neurons are combined. This process reduces the spatial dimensionality of the feature maps, which helps improve the model's resistance to overfitting. Maximum pooling is commonly employed in these layers. The integration and processing of features extracted by preceding layers are pivotal tasks performed by the fully connected layers, culminating in the generation of the architecture's final output.

The CNN model utilized in this research comprises four layers, as depicted in Figure 5. Firstly, the convolution layer processes the output from the embedding layer. Specifically, the input to the convolution layer is a 3D tensor, and the output retains the same shape. However, the dimensions of the output may vary depending on the number of filters and other parameters employed. The parameters for the CNN layer are detailed in Table 5.

*Table 5. CNN parameters*

| Parameter | Value |
|---|---|
| Filters | 128 |
| Kernel_size | 5 |
| Activation | Relu |
| Pooling type | Max pooling layer |
| Pool size | 5 |

The second layer is the pooling layer, specifically utilizing the Max Pooling technique. Max pooling is a common approach employed in CNNs for feature reduction and spatial hierarchies. It operates by down-sampling the output of the preceding Conv1D layer, thereby decreasing the spatial dimensions of the input data while preserving essential features. In essence, it extracts the most salient information from the feature maps. We set the size of the pooling window to 5, indicating that for every region of 5 consecutive elements in the input, the max pooling layer preserves only the maximum value. Max pooling is a straightforward operation: for each region of elements defined by the pool size, it selects the maximum value from that region and forwards it to the next layer. Suppose for the simple sentence "*Came here for dinner and had a fantastic experience*". Each word in this sentence represented using word vectors, and each word vector has a dimension of 5. Here are the word vectors: "*Came*": [0.2, 0.5, 0.3, 0.1, 0.7], "*here*": [0.7, 0.4, 0.2, 0.9, 0.6], "*for*": [0.4, 0.7, 0.1, 0.3, 0.2], "*dinner*": [0.9, 0.5, 0.1, 0.4, 0.2], "*and*": [0.3, 0.5, 0.1, 0.7, 0.2], "*had*": [0.1, 0.4, 0.6, 0.2, 0.7], "*a*": [0.2, 0.6, 0.1, 0.3, 0.8], "*fantastic*": [0.5, 0.1, 0.3, 0.4, 0.2], "*experience*" [0.6, 0.1, 0.2, 0.4, 0.3]. The max pooling operation setting with a filter size of 5. The filter, in this case, is a window of size 5 words that slides through the sentence. The filter starts at the beginning of the sentence: "*Came here for dinner and*": value [0.9, 0.7, 0.3, 0.9, 0.7] then move to the window one to the right "*here for dinner and had*": value [0.7, 0.7, 0.6, 0.9, 0.7] continue moving the window with "*for dinner and had a*": value [0.9, 0.7, 0.6, 0.7, 0.8]. It is continued until the end of the sentence. Now, maximum values collected from each position of the filter then slid it through the sentence. These maximum values are representative of the most important features within the filter's window. In this example, the extracted maximum values form a new sequence: [0.9, 0.7, 0.6, 0.9, 0.8]. This new sequence is a reduced-dimensional representation of the original text, capturing the most important information from the sentence. This sequence can use as input to further layers in neural network for

text classification. In practice, multiple filters can use with different window sizes to capture a range of features from the text, and the output of each filter is concatenated or combined before passing it through fully connected layers for classification. This allows the model to learn different levels of information from the text, from individual words to longer phrases or patterns. Max pooling helps in reducing the spatial dimensions of the data, reducing the number of parameters in the network and mitigating overfitting. It also helps in making the network translation-invariant, meaning it can recognize features in different parts of the input sequence regardless of their exact position. By retaining the maximum values, max pooling retains the most important features while discarding less significant details.

Following the pooling layer, the output is next transmitted to the flattened layer. The Flatten layer is a layer that is mostly utilized to transform the 2D or 3D data from the previous layer into a 1D vector. It preserves the number of elements in the data while converting it from a multidimensional form to a one-dimensional form. The Flatten layer is used to transform the output from the preceding layer, which could have several dimensions, into a single-dimensional vector. The final layer in our CNN model is the dense layer, which is also referred to as the completely linked layer or the output layer. Each neuron in this layer is intricately linked to every neuron in the preceding layer. The number of neurons in the layer dictates the configuration of the output from this layer. For our specific scenario, we have configured the dense layer to consist of only one neuron, and the chosen activation function is sigmoid. Hence, the resulting form of the dense layer is (batch_size, 1). The final one-dimensional output signifies the model's forecast for each input in the batch, with a numerical value ranging from 0 to 1 as a result of the sigmoid activation function. When the number is near 0, the model forecasts that the input has a negative sentiment, whereas a value close to 1 signifies a positive sentiment prediction. The structure of our suggested CNN model is depicted in Figure 6.

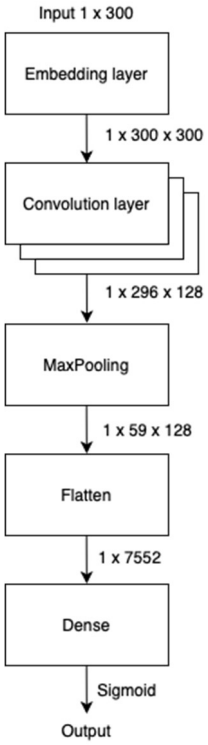


*Figure 6. Architecture of CNN Model*

### 3.3.2 Long Short-Term Memory (LSTM)

RNNs are frequently successful at analyzing sequences of data, particularly for problems involving lengthy text sequences. LSTM networks are a specific sort of deep neural network architecture that is built upon RNNs. Unlike traditional RNNs, which face difficulties in processing input sequences of any length because of the vanishing gradient problem, LSTM networks are explicitly intended to tackle this challenge. LSTMs utilize specialized units known as cells, which consist of devices like forget gates to keep or delete information as time progresses selectively. The gates, which consist of the input gate, output gate, and forget gate, control the movement of information within the LSTM cell. The input gate regulates the influx of fresh data into the cell, the forget gate decides which information to eliminate from the cell's memory, and the output gate manages the transmission of information from the cell's memory to the output. In addition, LSTM models facilitate the propagation of error signals over extended sequences by ensuring a consistent and steady flow of gradients throughout time. Sequential data models are capable of capturing long-term dependencies, which makes them very suitable for tasks such as sentiment analysis of text.

The gates in LSTM networks are responsible for determining whether information should be preserved and when it should be retrieved. LSTMs are a specific variant of RNNs that are specifically developed to tackle the issue of the vanishing or exploding gradient problem that is commonly faced by conventional RNNs. Like other types of RNNs, LSTM models generate output by considering input from the current time step as well as the output from the previous time step. An LSTM unit comprises a memory cell $c_t$, which retains its state during any period, and three non-linear gates: an input gate $i_t$, a forget gate $f_t$, an output gate $o_t$. The calculation formula is as follows, where $\sigma$ represents sigmoid function and $\odot$ symbolizes dot multiplication. These gates are specifically intended to control the flow of information into and out of the memory cell. The LSTM transition has been implemented using the equations provided below [102]:

The $i_t$ is the input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$

The $f_t$ is the forget gate:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (4)$$

The $o_t$ is the output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

The symbol $\tilde{c}_t$ represents the current condition of the candidate memory cell at the current time step. The function tanh refers to the tangent hyperbolic function:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (6)$$

The $c_t$ represents the current time state value stored in a memory cell, with the values of $f_t$ and $i_t$ ranging from 0 to 1. The computation of $i_t \odot \hat{C}_t$ determines the specific new information that is stored in $c_t$ from the candidate unit $\tilde{c}_t$ . The computation of $f_t \odot c_{t-1}$ determines which information is preserved and which is disregarded in the preceding memories $c_{t-1}$ [103].

$$c_t = i_t \odot \hat{C}_t + f_t \odot c_{t-1} \quad (7)$$

$h_t$ is the hidden layer state at time $t$ :

$$h_t = o_t \odot \tanh(c_t) \quad (8)$$

The input vector to the LSTM unit is denoted as $x_t$ the weight matrices are represented by $W$, and the bias vector parameters are represented by $b$.

### 3.3.3 Bidirectional LSTM (bi-LSTM)

The LSTM primarily considers the historical information in a sequence, which can sometimes be insufficient. Gaining access to future information, just as it accesses past information, could significantly enhance performance in sequence-based tasks. A Bi-LSTM consists of a forward LSTM layer, which captures historical information, and a backward LSTM layer, which captures future information. Both layers are connected to the same output layer [103]. The key advantage of this architecture is that it provides a comprehensive view of the sequence context by integrating both past and future information. Let's consider the input of time $t$ is the word embedding $x_t$, at time $t-1$, the output of the forward hidden unit is $\overleftarrow{h}_{t-1}$, then the output of the hidden unit at time $t$ is equal as follow:

$$\overleftarrow{h}_t = L(x_t, \vec{h}_{t-1}, c_{t-1}) \quad (9)$$

$$\vec{h}_t = L(x_t, \overleftarrow{h}_{t-1}, c_{t-1}) \quad (10)$$

Where $L$ denotes the hidden layer operation of the. LSTM hidden layer. The forward output vector is $\vec{h}_t \in R^{1xH}$ and the backward output vector is $\overleftarrow{h}_t \in R^{1xH}$ , and they should be combined to obtain the text feature.

### 3.3.4 Gated Recurrent Units (GRU)

Gated Recurrent Units (GRUs) are a kind of RNNs that demonstrate comparable empirical outcomes to LSTM networks, but with a simpler structure. A standard GRU design comprises of two gates: the reset gate and the update gate. This architectural design incorporates a reduced number of parameters. In the GRU architecture, transitions are carried out based on the equations given below [102]:

The $z_t$ is the update gate:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (11)$$

The $r_t$is the reset gate:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (12)$$

The $\hat{h}_t$ is the current memory content:

$$\hat{h}_t = \tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h) \quad (13)$$

The $h_t$ is the output gate:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (14)$$

Where $x_t$ denotes the input vector, $h_t$ denotes the output vector, $r_t$ corresponds to the reset gate vector, $z_t$ corresponds to the update gate vector and $W$, $U$, and $b$ corresponds to parameter matrices and vector.

### 3.3.5 Bidirectional Gated Recurrent Units (bi-GRU)

Models that possess a bi-directional structure are capable of acquiring knowledge from both preceding and succeeding data while analyzing present data. The bi-GRU model is characterized by the states of two GRUs, each functioning unidirectionally in opposite directions [104]. One GRU progresses in a forward direction, commencing from the initial point of the data sequence, while the other GRU progresses in a backward direction. This configuration enables the integration of information from both preceding and subsequent periods to impact the present conditions [105]. The bi-GRU is defined in the following manner:

$$\vec{h}_t = GRU_{fwd}(x_t, \vec{h}_{t-1}) \quad (15)$$

$$\overleftarrow{h}_t = GRU_{bwd}(x_t, \overleftarrow{h}_{t-1}) \quad (16)$$

$$h_t = \vec{h}_t \oplus \overleftarrow{h}_t \quad (17)$$

Where $\vec{h}_t$ is the state of the forward GRU, $\overleftarrow{h}_t$ is the state of the backward GRU, $\oplus$ indicates the operation of concatenating two vectors. The architecture RNN model (LSTM, bi-LSTM, GRU, bi-GRU) that we use in this study can be seen in Figure 7. We set the dimensions for all RNN models to be the same, namely 64.

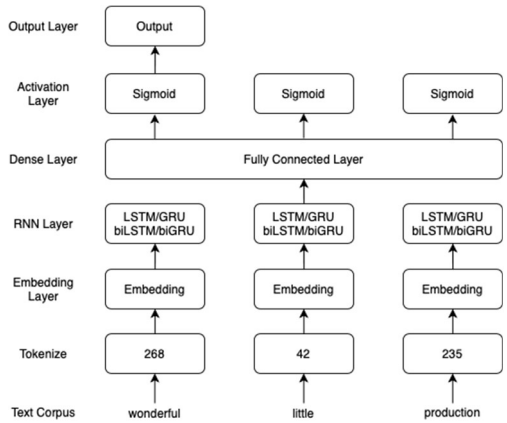


*Figure 7. Architecture of RNN Model*

## 4. METHODS

### 4.1 Dataset

We conduct experiments on various online review datasets to study the impact of word embeddings across different datasets. We have selected three benchmark classification datasets that vary in average sample length. Statistics from the datasets used in this study can be found in Table 6.

*Table 6. Dataset Statistics*

| Dataset | Positive | Negative | Total |
|---|---|---|---|
| IMDB | 25000 | 25000 | 50000 |
| Amazon | 20229 | 19771 | 40000 |
| Yelp | 299000 | 299000 | 598000 |

#### 4.1.1 IMDB Dataset

The IMDb dataset contains 50,000 movie reviews for text analytics [106]. Reviews are classified as positive or negative based on the IMDb rating system, making it a benchmark dataset for sentiment classification. This large dataset features full-length reviews, and the task involves determining whether the movie reviews are positive or negative. The data includes two columns: the review column and the sentiment column. The review column contains the online reviews, while the sentiment column indicates the polarity of these reviews based on their content. In this dataset, we convert the polarity in the sentiment column into a binary class; specifically, we change 'positive' to 1 and 'negative' to 0. This conversion facilitates the processing of the data into our proposed model. We divided the 50,000 entries into 40,000 for training and 10,000 for testing. From the 40,000 training entries, we allocated 20%, or 8,000 entries, for validation. The IMDb dataset we used is balanced, with an equal number of positive and negative classes.

#### 4.1.2 Amazon Dataset

Amazon is an American multinational technology company with business interests that include e-commerce. In its e-commerce operations, Amazon purchases and stores inventory, handling everything from shipping and pricing to customer service and returns. It is one of the largest e-commerce platforms, renowned for its vast number of customer reviews. In this study, we chose to use Amazon customer product reviews. The total dataset consists of 40,000 entries, divided into 36,000 for training and 4,000 for testing. We allocated 20% of the training data, or 7,200 entries, for validation purposes. The Amazon dataset we used in this study

is unbalanced, with an unequal number of positive and negative reviews.

#### 4.1.2 Yelp Dataset

The Yelp review dataset comprises binary sentiment classification data. For this study, we utilize a collection of 560,000 highly polar Yelp reviews for training, with an additional 38,000 reserved for testing. This dataset is sourced from the Yelp Dataset Challenge 2015 data and is based on the dataset constructed by Zhang et al. [107]. The construction categorizes reviews with 1 or 2 stars as negative and those with 3 or 4 stars as positive. Each polarity includes 280,000 training samples and 19,000 testing samples, randomly selected. Overall, there are 560,000 training samples and 38,000 testing samples. The negative polarity is labeled as class 0, while the positive polarity is labeled as class 1. To create a validation dataset, 20% of the training data, or 112,000 entries, is set aside.

### 4.2 Performance Matrix

The model evaluation criteria utilized in this work are accuracy, precision, recall, F1 score, and the Matthews Correlation Coefficient (MCC), which align with the metrics adopted in other studies. The computation parameters are specified as follows:

1) TP: stand for True Positive is the number of comments that classify favorable merchandise remarks as positive.
2) FP: stand for False Positive is the number of comments that incorrectly label unfavorable product feedback as positive.
3) TN: stand for True Negative refers to the count of negative comments that have been correctly categorized as negative comments.
4) FN: stand for False Negative is the number of comments that classify positive ratings of products as negative.
5) Accuracy refers to the proportion of comments that have been accurately predicted out of the total number of comments.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (18)$$

6) Precision refers to the proportion of accurately anticipated positive remarks out of all the expected positive comments.

$$precision = \frac{TP}{TP + FN} \quad (19)$$

7) Recall refers to the proportion of positive comments that were accurately anticipated out of all the comments in the actual class.

$$recall = \frac{TP}{TP + FN} \quad (20)$$

8) F1 score refers to the arithmetic mean of precision and recall, where precision is the ratio of true positive predictions to the sum of true positive and false positive predictions, and recall is the ratio of true positive predictions to the sum of true positive and false negative predictions.

$$F1 = \frac{2 * precision * recall}{precision + recall} \quad (21)$$

9) MCC refers to the correlation coefficient measuring the relationship between the observed and expected binary categorization.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(FN + TN)(FP + TN)(TP + FN)}} \quad (22)$$

### 4.2 Result Analysis and Discussion

In the initial phase, we conducted various experiments to identify the most effective model for predicting sentiment polarity in online reviews. These experiments involved comparing the efficiency of different feature engineering methods, including FastText, Glove, word2vec, Bert, and Albert. The algorithms employed in our experiments encompassed CNN, LSTM, bi-LSTM, GRU, and bi-GRU. The first experiment focused on the IMDB dataset, which is balanced dataset. The outcomes for the IMDB dataset in context-independent embedding are detailed in Table 7. Upon a comprehensive analysis, it is evident that, for each feature engineering model, the bi-GRU model consistently demonstrated the best performance. Specifically, for fastText, word2vec, and Glove, the Matthews Correlation Coefficient (MCC) results were 74.77%, 77.4%, and 74.54%, respectively.
The outcomes vary when examined from the perspective of each deep learning model. Specifically, for the CNN model, optimal performance, at 73.22%, is achieved when paired with fastText. In the case of the bi-LSTM model, the

best performance, reaching 73.21%, is observed with the Glove embedding. Meanwhile, for the LSTM, GRU, and bi-GRU models, the highest performance is consistently attained when utilizing the word2vec feature engineering model, yielding respective results of 74.73%, 76.22%, and 77.4%. Upon an overall analysis of the experimental results, the word2vec and bi-GRU models exhibit the best performance, both achieving an MCC value of 77.4%.

Meanwhile, in experiments utilizing transformers as embedding layers, it is evident that both Bert and Albert in IMDB dataset achieve their highest performance when integrated with the GRU model, yielding MCC values of 74.81% and 74.79% for Bert and Albert, respectively. Comparing Bert and Albert's performance on the GRU model, the differences observed are not particularly significant; in fact, both exhibit identical F1 Scores. The comprehensive performance results for Bert and Albert are presented in Table 9. However, when considering the overall performance, the most favorable outcomes were achieved with the word2vec and biGRU models.

The second experiment was conducted on the Amazon dataset, characterized by its unbalanced dataset. The experimental results for the Amazon dataset are outlined in Table 8. Notably, across the three feature engineering approaches, the optimal performance is consistently achieved when employing the bi-GRU model, with MCC values of 71.69%, 74.48%, and 72.4%, respectively. This trend extends to the deep learning models as well, where all five models exhibit their best performance when utilizing the word2vec feature engineering. Upon a comprehensive comparison of the results, the highest performance is observed when word2vec is paired with bi-GRU, attaining an MCC value of 74.48%. These results are similar to the results obtained in the IMDB dataset.

In the experiment utilizing transformers for the Amazon dataset, unlike the previous dataset, Bert achieved its highest performance of 87.13% when combined with GRU. In contrast, Albert achieved its highest performance of 87.1% when combined with the BiGRU model. The comprehensive performance results for Bert and Albert are presented in Table 10.

Let's now turn our attention to the third dataset, namely the Yelp dataset. Unlike the previous datasets, Yelp is balanced but boasts a larger dataset size. The experimental results for the Yelp dataset using context-independent embedding can be seen in Table 11. Notably, performance varies across feature engineering models. For the fastText model, optimal performance is achieved when paired with the GRU model, reaching 87.55%. Conversely, word2vec and Glove exhibit their best performance when combined with bi-LSTM, attaining values of 88.55% and 86.99%, respectively. From the perspective of deep learning models, all five models obtained the best performance when utilizing word2vec. In contrast to the trends observed in the previous datasets, where word2vec paired with bi-GRU yielded the best performance, the Yelp dataset achieves its best performance with word2vec and bi-LSTM models, reaching an MCC score of 88.55%.

From Table 12. we can see that the performance obtained by employing transformers on the Yelp dataset indicates that Bert and Albert achieve their optimal results when paired with GRU, with MCC values of 89.28% and 88.68% respectively. Upon examination, Bert outperforms Albert.

## 5. CONCLUSION AND FUTURE WORK

This paper aimed to evaluate different deep learning models and different feature engineering models to predict the sentiment polarity of textual online review of three different datasets with different domain and different amount of data. Five different variations of feature extraction model we used, fastText, word2vec, Glove, Bert, and Albert then compared concerning five deep learning methods: CNN, LSTM, bi-LSTM, GRU, and bi-GRU. Word embedding plays a crucial role in text classification by transforming text into vectorized numerical representations which allows us to use it as an input to the machine learning algorithm.

The first dataset is the IMDB dataset which is a balanced dataset. The best performance obtained when using the context-independent embedding approach is when using word2vec and bi-GRU, namely with an MCC value of 77.4%. Meanwhile, when using transformers, the best performance was obtained when using Bert and GRU with an MCC value of 74.81%. For the second dataset, namely the Amazon dataset, just like the IMDB dataset, the best performance was obtained when using the word2vec and bi-GRU models, namely 74.48%. Meanwhile, for the transformer-based model, the best performance was obtained when using the Bert and GRU models, namely with an MCC value of 87.13%. For the third dataset, namely Yelp, the word2vec and bi-LSTM models demonstrated the best performance, achieving an impressive 88.55%. Meanwhile, for the transformer-based model, the best performance was obtained when using the Bert and GRU models, namely with an MCC value of 89.28%.

Based on the comprehensive results, it can be asserted that feature extraction based on context-

independent embedding yielding the best performance in sentiment classification is word2vec. Word2vec consistently excels for both balanced and unbalanced datasets. Regarding the optimal deep learning model, it is evident that bi-GRU outperforms others for the IMDB and Amazon datasets, while bi-LSTM proves superior for the larger Yelp dataset. Meanwhile, for transformer-based embedding, the three datasets have the same results, namely getting the best performance using Bert and GRU. Overall, when analyzed from the perspective of word embeddings, Bert demonstrates superiority on the Amazon and Yelp datasets, whereas word2vec outperforms Bert on the IMDB dataset. Based on our preliminary analysis, this variance may be attributed to the distinct nature of the test data within the IMDB dataset compared to the other two datasets.

In summary, word2vec and Bert emerges as the preferred feature extraction model for addressing sentiment classification tasks, bi-GRU and GRU deep learning model stands out as superior among the tested models. Notably, Glove consistently yields the lowest performance across all model schemes and datasets. This research serves as a foundation for future research. The results obtained require additional processing, including a hyperparameter tuning process such as the length of the input sentence and the number of iterations of the model on the performance of the model to analyze performance patterns for each model scheme. Furthermore, research and experiments need to be carried out on Albert, which is smaller and lighter than Bert, in order to create a lighter and faster model. Apart from that, there will be a process of using more than one type of embedding for the deep learning model to obtain even better performance.

## REFERENCES:

[1] P. Chauhan, N. Sharma, and G. Sikka, "The emergence of social media data and sentiment analysis in election prediction," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 2, pp. 2601–2627, 2021.

[2] R. K. Behera, M. Jena, S. K. Rath, and S. Misra, "Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data," *Inf. Process. Manag.*, vol. 58, no. 1, p. 102435, 2021.

[3] L. Yang, Y. Li, J. Wang, and R. S. Sherratt, "Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning," *IEEE Access*, vol. 8, pp. 23522–23530, 2020.

[4] D. Zeng, Y. Dai, F. Li, J. Wang, and A. K. Sangaiah, "Aspect based sentiment analysis by a linguistically regularized CNN with gated mechanism," *J. Intell. Fuzzy Syst.*, vol. 36, no. 5, pp. 3971–3980, 2019.

[5] K. W. Trisna and H. J. Jie, "Deep Learning Approach for Aspect-Based Sentiment Classification: A Comparative Review," *Appl. Artif. Intell.*, vol. 36, no. 1, 2022.

[6] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 23, pp. 1–12, 2021.

[7] I. Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, and E. Cambria, "Bayesian network based extreme learning machine for subjectivity detection," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1780–1797, 2018.

[8] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Eng. J.*, vol. 5, no. 4, pp. 1093–1113, 2014.

[9] H. Gao, X. Zeng, and C. Yao, "Application of improved distributed naive Bayesian algorithms in text classification," *J. Supercomput.*, vol. 75, no. 9, pp. 5831–5847, 2019.

[10] S. Dey, S. Wasif, D. S. Tonmoy, S. Sultana, J. Sarkar, and M. Dey, "A Comparative Study of Support Vector Machine and Naive Bayes Classifier for Sentiment Analysis on Amazon Product Reviews," *2020 Int. Conf. Contemp. Comput. Appl. IC3A 2020*, pp. 217–220, 2020.

[11] K. Shah, H. Patel, D. Sanghvi, and M. Shah, "A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification," *Augment. Hum. Res.*, vol. 5, no. 1, 2020.

[12] I. C. Chang, T. K. Yu, Y. J. Chang, and T. Y. Yu, "Applying text mining, clustering analysis, and latent dirichlet allocation techniques for topic classification of environmental education journals," *Sustain.*, vol. 13, no. 19, 2021.

[13] Z. Islam, J. Liu, J. Li, L. Liu, and W. Kang, "A Semantics Aware Random Forest for Text Classification," *CIKM 2019 - Proc. 2019 ACM Int. Conf. Inf. Knowl. Manag.*, pp. 1061–1070, 2019.

[14] T. Georgieva-Trifonova and M. Duraku, "Research on N-grams feature selection methods for text classification," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1031, no. 1, 2021.

[15] L. Wang, X. kang Wang, J. juan Peng, and J.

qiang Wang, "The differences in hotel selection among various types of travellers: A comparative analysis with a useful bounded rationality behavioural decision support model," *Tour. Manag.*, vol. 76, no. October 2018, p. 103961, 2020.

[16] L. Dang, C. Wang, H. Han, and Y. E. Hou, "A Hybrid BiLSTM-ATT Model for Improved Accuracy Sentiment Analysis," *Proc. - 24th IEEE Int. Conf. High Perform. Comput. Commun. 8th IEEE Int. Conf. Data Sci. Syst. 20th IEEE Int. Conf. Smart City 8th IEEE Int. Conf. Dep*, pp. 2182–2188, 2022.

[17] R. M. Samant, M. R. Bachute, S. Gite, and K. Kotecha, "Framework for Deep Learning-Based Language Models Using Multi-Task Learning in Natural Language Understanding: A Systematic Literature Review and Future Directions," *IEEE Access*, vol. 10, pp. 17078–17097, 2022.

[18] J. Xiong, D. Yu, S. Liu, L. Shu, X. Wang, and Z. Liu, "A review of plant phenotypic image recognition technology based on deep learning," *Electron.*, vol. 10, no. 1, pp. 1–19, 2021.

[19] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Res.*, vol. 43, no. 4, pp. 244–252, 2019.

[20] J. Wang, R. Chen, and Z. He, "Traffic speed prediction for urban transportation network: A path based deep learning approach," *Transp. Res. Part C Emerg. Technol.*, vol. 100, no. January, pp. 372–385, 2019.

[21] Y. Tu, Y. Lin, J. Wang, and J. U. Kim, "Semi-supervised learning with generative adversarial networks on digital signal modulation classification," *Comput. Mater. Contin.*, vol. 55, no. 2, pp. 243–254, 2018.

[22] G. Lăzăroiu, M. Andronie, M. Iatagan, M. Geamănu, R. Ștefănescu, and I. Dijmărescu, "Deep Learning-Assisted Smart Process Planning, Robotic Wireless Sensor Networks, and Geospatial Big Data Management Algorithms in the Internet of Manufacturing Things," *ISPRS Int. J. Geo-Information*, vol. 11, no. 5, 2022.

[23] M. Premkumar and T. V. P. Sundararajan, "DLDM: Deep learning-based defense mechanism for denial of service attacks in wireless sensor networks," *Microprocess. Microsyst.*, vol. 79, no. September, p. 103278, 2020.

[24] M. Sun, I. Konstantelos, and G. Strbac, "A Deep Learning-Based Feature Extraction Framework for System Security Assessment," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5007–5020, 2018.

[25] D. Zeng, Y. Dai, F. Li, R. S. Sherratt, and J. Wang, "Adversarial learning for distant supervised relation extraction," *Comput. Mater. Contin.*, vol. 55, no. 1, pp. 121–136, 2018.

[26] U. A. Chauhan, M. T. Afzal, A. Shahid, M. Abdar, M. E. Basiri, and X. Zhou, "A comprehensive analysis of adverb types for mining user sentiments on amazon product reviews," *World Wide Web*, vol. 23, no. 3, pp. 1811–1829, 2020.

[27] B. Liu and L. Zhang, *A Survey of Opinion Mining and Sentiment Analysis*. Mining Text Data Springer, 2012.

[28] W. Zhao, H. Peng, S. Eger, E. Cambria, and M. Yang, "Towards scalable and reliable capsule networks for challenging NLP applications," *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, pp. 1549–1559, 2020.

[29] A. Duric and F. Song, "Feature selection for sentiment analysis based on content and syntax models," *Decis. Support Syst.*, vol. 53, no. 4, pp. 704–711, 2012.

[30] A. Abbasi, S. France, Z. Zhang, and H. Chen, "Selecting attributes for sentiment classification using feature relation networks," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 3, pp. 447–462, 2011.

[31] S. Poria, I. Chaturvedi, E. Cambria, and F. Bisio, "Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2016-Octob, pp. 4465–4473, 2016.

[32] I. Chaturvedi, Y. S. Ong, I. W. Tsang, R. E. Welsch, and E. Cambria, "Learning word dependencies in text by means of a deep recurrent belief network," *Knowledge-Based Syst.*, vol. 108, pp. 144–154, 2016.

[33] M. Ehsan Basiri and A. Kabiri, "Words are important: Improving sentiment analysis in the Persian language by lexicon refining," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 17, no. 4, 2018.

[34] A. Srivastava, V. Singh, and G. S. Drall, "Sentiment Analysis of Twitter Data," *Int. J. Healthc. Inf. Syst. Informatics*, vol. 14, no. 2, pp. 1–16, 2019.

[35] S. J. Jolly and G. W. Gramenz, "Customizing a Norm-Referenced Achievement Test to

Achieve Curricular Validity: A Case Study," *Educ. Meas. Issues Pract.*, vol. 3, no. 3, pp. 16–18, 1984.

[36] K. Machová, M. Mikula, X. Gao, and M. Mach, "Lexicon-based sentiment analysis using particle swarm optimization," *Electron.*, vol. 9, no. 8, pp. 1–22, 2020.

[37] F. Wunderlich and D. Memmert, "Innovative approaches in sports science-Lexicon-based sentiment analysis as a tool to analyze sports-related twitter communication," *Appl. Sci.*, vol. 10, no. 2, 2020.

[38] S. Taj, B. B. Shaikh, and A. Fatemah Meghji, "Sentiment analysis of news articles: A lexicon based approach," *2019 2nd Int. Conf. Comput. Math. Eng. Technol. iCoMET 2019*, pp. 1–5, 2019.

[39] S. Baccianella, A. Esuli, and F. Sebastiani, "SENTIWORDNET 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," *Proc. 7th Int. Conf. Lang. Resour. Eval. Lr. 2010*, vol. 0, pp. 2200–2204, 2010.

[40] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," *HLT/EMNLP 2005 - Hum. Lang. Technol. Conf. Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, no. October, pp. 347–354, 2005.

[41] J. W. Pennebaker, M. R. Mehl, and K. G. Niederhoffer, "Psychological Aspects of Natural Language Use: Our Words, Our Selves," *Annu. Rev. Psychol.*, vol. 54, pp. 547–577, 2003.

[42] M. E. Basiri and A. Kabiri, "HOMPer: A new hybrid system for opinion mining in the Persian language," *J. Inf. Sci.*, vol. 46, no. 1, pp. 101–117, 2020.

[43] F. Amiri, S. Scerri, and M. H. Khodashahi, "Lexicon-based sentiment analysis for Persian text," *Int. Conf. Recent Adv. Nat. Lang. Process. RANLP*, vol. 2015-Janua, pp. 9–16, 2015.

[44] E. Cambria, "An introduction to concept-level sentiment analysis," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8266 LNAI, no. PART 2, pp. 478–483, 2013.

[45] B. Agarwal and N. Mittal, *Prominent Feature Extraction for Sentiment Analysis*. 2016.

[46] G. Yoo and J. Nam, "A hybrid approach to sentiment analysis enhanced by sentiment lexicons and polarity shifting devices," *13th Work. Asian Lang. Resour.*, pp. 21–28, 2018.

[47] S. Jameel, Z. Bouraoui, and S. Schockaert, "Unsupervised learning of distributional relation vectors," *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 23–33, 2018.

[48] J. Yoon and H. Kim, "Multi-channel lexicon integrated CNN-BILSTM models for sentiment analysis," *Proc. 29th Conf. Comput. Linguist. Speech Process. ROCLING 2017*, pp. 244–253, 2017.

[49] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *52nd Annu. Meet. Assoc. Comput. Linguist. ACL 2014 - Proc. Conf.*, vol. 1, pp. 655–665, 2014.

[50] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *NAACL HLT 2015 - 2015 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc. Conf.*, no. 2011, pp. 103–112, 2015.

[51] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Syst. Appl.*, vol. 117, pp. 139–147, 2019.

[52] T. Demeester, T. Rocktäschel, and S. Riedel, "Distributed Representations of Words and Phrases and their Compositionality," *Adv. Neural Inf. Process. Syst.*, pp. 3111–3119, 2013.

[53] J. Pennington, R. Socher, and C. D. Manning, "GloVe : Global Vectors for Word Representation," *EMNLP*, pp. 1532–1543, 2014.

[54] C. Dos Santos and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," *Proc. Cool. 2014, 25th Int. Conf. Comput. Lingustics Tech. Pap.*, pp. 69–78, 2014.

[55] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv Prepr. arXiv1408.5882*, 2014.

[56] J. Wang, L. C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model," *54th Annu. Meet. Assoc. Comput. Linguist. ACL 2016 - Short Pap.*, pp. 225–230, 2016.

[57] H. Zhao, Z. Lu, and P. Poupart, "Self-adaptive hierarchical sentence model," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2015-Janua, pp. 4069–4076, 2015.

[58] D. Hyun, C. Park, M. C. Yang, I. Song, J. T.

Lee, and H. Yu, "Target-aware convolutional neural network for target-level sentiment analysis," *Inf. Sci. (Ny).*, vol. 491, pp. 166–178, 2019.

[59] S. Hochreiter, "Long Short-Term Memory," *Neural Comput.*, vol. 1780, pp. 1735–1780, 1997.

[60] Y. Mehta, N. Majumder, A. Gelbukh, and E. Cambria, "Recent trends in deep learning based personality detection," *Artif. Intell. Rev.*, vol. 53, no. 4, pp. 2313–2339, 2020.

[61] J. Xu, D. Chen, X. Qiu, and X. Huang, "Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification," *Arvix Prepr.*, no. arXiv:1610.04989, 2016.

[62] R. Moraes, J. F. Valiati, and W. P. Gavião Neto, "Document-level sentiment classification: An empirical comparison between SVM and ANN," *Expert Syst. Appl.*, vol. 40, no. 2, pp. 621–633, 2013.

[63] A. Chatterjee, U. Gupta, M. K. Chinnakotla, R. Srikanth, M. Galley, and P. Agrawal, "Understanding Emotions in Text Using Deep Learning and Big Data," *Comput. Human Behav.*, vol. 93, no. April 2018, pp. 309–317, 2019.

[64] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-Specific Word Embedding," *Acl*, pp. 1555–1565, 2014.

[65] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," *COLING 2016 - 26th Int. Conf. Comput. Linguist. Proc. COLING 2016 Tech. Pap.*, pp. 3298–3307, 2016.

[66] X. Zhu, P. Sobhani, and H. Guo, "Long short-term memory over recursive structures," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 2, pp. 1604–1612, 2015.

[67] A. E. D. Mousa and B. Schuller, "Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis," *15th Conf. Eur. Chapter Assoc. Comput. Linguist. EACL 2017 - Proc. Conf.*, vol. 2, pp. 1023–1032, 2017.

[68] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-Term memory networks," *ACL-IJCNLP 2015 - 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. Asian Fed. Nat. Lang. Process. Proc. Conf.*, vol. 1, pp. 1556–1566, 2015.

[69] Y. Ma, H. Peng, T. Khan, and E. Cambria, "Sentic LSTM : a Hybrid Network for Targeted Aspect-Based Sentiment Analysis," *Congnitive Comput.*, vol. 10, pp. 639–650, 2018.

[70] H. Chen, S. Li, P. Wu, N. Yi, S. Li, and X. Huang, "Fine-grained sentiment analysis of Chinese reviews using LSTM network," *J. Eng. Sci. Technol. Rev.*, vol. 11, no. 1, pp. 174–179, 2018.

[71] S. Wen *et al.*, "Memristive LSTM Network for Sentiment Analysis," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 51, no. 3, pp. 1794–1804, 2021.

[72] F. Hu, L. Li, Z. L. Zhang, J. Y. Wang, and X. F. Xu, "Emphasizing Essential Words for Sentiment Classification Based on Recurrent Neural Networks," *J. Comput. Sci. Technol.*, vol. 32, no. 4, pp. 785–795, 2017.

[73] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, 2005.

[74] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN," *Expert Syst. Appl.*, vol. 72, pp. 221–230, 2017.

[75] X. Niu, Y. Hou, and P. Wang, "Bi-Directional LSTM with Quantum Attention Mechanism for Sentence Modeling," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10635 LNCS, pp. 178–188, 2017.

[76] Z. Yang, D. Yang, D. Chris, X. He, A. Smola, and E. Hovey, "Hierarchical attention networks for document classification," *ArXiv*, pp. 1480–1489, 2016.

[77] L. Zhang, Y. Zhou, X. Duan, and R. Chen, "A Hierarchical multi-input and output Bi-GRU Model for Sentiment Analysis on Customer Reviews," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 322, no. 6, 2018.

[78] L. Li, L. Yang, and Y. Zeng, "Improving Sentiment Classification of Restaurant Reviews with Attention-Based Bi-GRU Neural Network," *Symmetry (Basel).*, 2021.

[79] Y. Pan and M. Liang, "Chinese Text Sentiment Analysis Based on BI-GRU and Self-attention," *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, no. Itnec, pp. 1983–1988, 2020.

[80] L. xia Luo, "Network text sentiment analysis method combining LDA text representation and GRU-CNN," *Pers. Ubiquitous Comput.*, vol. 23, no. 3–4, pp. 405–412, 2019.

[81] R. Ni and H. Cao, "Sentiment Analysis based on GloVe and LSTM-GRU," *Chinese Control Conf. CCC*, vol. 2020-July, pp. 7492–7497, 2020.

[82] N. Aslam, F. Rustam, E. Lee, P. B. Washington, and I. Ashraf, "Sentiment Analysis and Emotion Detection on Cryptocurrency Related Tweets Using Ensemble LSTM-GRU Model," *IEEE Access*, vol. 10, pp. 39313–39324, 2022.

[83] Y. Cheng *et al.*, "Sentiment Analysis Using Multi-Head Attention Capsules with Multi-Channel CNN and Bidirectional GRU," *IEEE Access*, vol. 9, pp. 60383–60395, 2021.

[84] C. Wang, P. Nulty, and D. Lillis, "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," *ACM Int. Conf. Proceeding Ser.*, pp. 37–46, 2020.

[85] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, pp. 1–25, 2018.

[86] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017.

[87] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in Translation: Contextualized Word Vectors," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017, pp. 6294–6305.

[88] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," *COLING 2018 - 27th Int. Conf. Comput. Linguist. Proc.*, pp. 1638–1649, 2018.

[89] M. E. Peters *et al.*, "Deep contextualized word representations," *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, pp. 2227–2237, 2018.

[90] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[91] M. C. Kenton, L. Kristina, and J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805*, 2019.

[92] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: a Lite Bert for Self-Supervised Learning of Language Representations," *8th Int. Conf. Learn. Represent. ICLR 2020*, pp. 1–17, 2020.

[93] W. Huang *et al.*, "Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach," in *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, 2019, pp. 1051–1060.

[94] X. Zhang, J. Zhao, and Y. Lecun, "Character-level convolutional networks for text classification," *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 649–657, 2015.

[95] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM Neural Network for Text Classification," *arXiv Prepr. arXiv1511.08630*, 2015.

[96] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," *COLING 2016 - 26th Int. Conf. Comput. Linguist. Proc. COLING 2016 Tech. Pap.*, vol. 2, no. 1, pp. 3485–3495, 2016.

[97] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: Compressing text classification models," pp. 1–13, 2016.

[98] I. Santos, N. Nedjah, and L. De Macedo Mourelle, "Sentiment analysis using convolutional neural network with fasttext embeddings," *2017 IEEE Lat. Am. Conf. Comput. Intell. LA-CCI 2017 - Proc.*, vol. 2018-Janua, pp. 2–6, 2017.

[99] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

[100] P. X. V. Nguyen, T. T. T. Hong, K. Van Nguyen, and N. L. T. Nguyen, "Deep Learning versus Traditional Classifiers on Vietnamese Students' Feedback Corpus," *NICS 2018 - Proc. 2018 5th NAFOSTED Conf. Inf. Comput. Sci.*, pp. 75–80, 2019.

[101] A. Onan, "Mining opinions from instructor evaluation reviews: A deep learning approach," *Comput. Appl. Eng. Educ.*, vol. 28, no. 1, pp. 117–138, 2020.

[102] L. M. Rojas-Barahona, "Deep learning for sentiment analysis," *Lang. Linguist. Compass*, vol. 10, no. 12, pp. 701–719, 2016.

[103] F. Long, K. Zhou, and W. Ou, "Sentiment

analysis of text based on bidirectional LSTM with multi-head attention," *IEEE Access*, vol. 7, pp. 141960–141969, 2019.

[104] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 5, no. 2015, pp. 3574–3583, 2016.

[105] X. Liu, Y. Wang, X. Wang, H. Xu, C. Li, and X. Xin, "Bi-directional gated recurrent unit neural network based nonlinear equalizer for coherent optical communication system," *Opt. Express*, vol. 29, no. 4, p. 5923, 2021.

[106] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," *ACL-HLT 2011 - Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol.*, vol. 1, pp. 142–150, 2011.

[107] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *Adv. Neural Inf. Process. Syst.*, pp. 649–657, 2015.

*Table 7. Results of IMDB Dataset using context-independent embedding*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| FastText + CNN | 86,61 | 86,22 | 87,4 | 86,8 | 73,22 |
| FastText + LSTM | 85,8 | 86,45 | 85,18 | 85,81 | 71,61 |
| FastText + biLSTM | 86,02 | 87,79 | 83,93 | 85,82 | 72,12 |
| FastText + GRU | 86,39 | 83,11 | 91,61 | 87,15 | 73,15 |
| **FastText + biGRU** | **87,38** | **88,13** | **86,62** | **87,37** | **74,77** |
| word2vec + CNN | 86,51 | 85,96 | 87,52 | 86,73 | 73,03 |
| word2vec + LSTM | 87,34 | 86,01 | 89,42 | 87,68 | 74,73 |
| word2vec + biLSTM | 86,12 | 84,78 | 88,31 | 86,51 | 72,29 |
| word2vec + GRU | 87,98 | 91,37 | 84,08 | 87,58 | 76,22 |
| **word2vec + biGRU** | **88,7** | **88,9** | **88,65** | **88,77** | **77,4** |
| Glove + CNN | 84,32 | 84,52 | 84,32 | 84,42 | 68,64 |
| Glove + LSTM | 85,19 | 80,28 | 93,61 | 86,43 | 71,35 |
| Glove + biLSTM | 86,58 | 88 | 84,96 | 86,45 | 73,21 |
| Glove + GRU | 85,9 | 81,27 | 93,59 | 87 | 72,62 |
| **Glove + biGRU** | **87,18** | **84,78** | **90,87** | **87,72** | **74,54** |

*Table 8. Results of IMDB dataset using transformers*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| Bert + CNN | 87,11 | 84,36 | 91,35 | 87,71 | 74,47 |
| Bert + LSTM | 86,7 | 83,63 | 91,5 | 87,4 | 73,71 |
| Bert + biLSTM | 87,36 | 86,87 | 88,25 | 87,55 | 74,72 |
| **Bert + GRU** | **87,4** | **86,68** | **88,61** | **87,63** | **74,81** |
| Bert + biGRU | 87,2 | 87,88 | 86,52 | 87,2 | 74,41 |
| albert +CNN | 86,14 | 83,23 | 90,8 | 86,85 | 72,56 |
| albert + LSTM | 87,08 | 87,86 | 86,29 | 87,06 | 74,17 |
| albert + biLSTM | 87,03 | 86,51 | 87,97 | 87,24 | 74,07 |
| **albert + GRU** | **87,39** | **86,65** | **88,63** | **87,63** | **74,79** |
| albert + biGRU | 87,09 | 87,89 | 86,27 | 87,07 | 74,19 |

*Table 9. Results of Amazon dataset using context-independent embedding*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| FastText + CNN | 85,15 | 88,39 | 81,75 | 84,94 | 70,55 |
| FastText + LSTM | 84,95 | 86,26 | 83,26 | 85 | 69,97 |
| FastText + biLSTM | 85,52 | 84,74 | 87,51 | 86,1 | 71,05 |
| FastText + GRU | 85,4 | 82 | 91,61 | 86,54 | 71,21 |
| **FastText + biGRU** | **85,8** | **84,01** | **89,26** | **86,56** | **71,69** |
| word2vec + CNN | 86,67 | 88,2 | 83,16 | 85,61 | 71,5 |
| word2vec + LSTM | 86,95 | 86,69 | 88,04 | 87,36 | 73,88 |
| word2vec + biLSTM | 85,65 | 90,68 | 80,23 | 85,14 | 71,87 |
| word2vec + GRU | 85,97 | 81,03 | 94,83 | 87,38 | 72,94 |
| **word2vec + biGRU** | **87,2** | **85,43** | **90,43** | **87,86** | **74,48** |
| Glove + CNN | 82,45 | 81,52 | 85,02 | 83,23 | 64,9 |
| Glove + LSTM | 85 | 87,21 | 82,87 | 84,98 | 70,11 |
| Glove + biLSTM | 84,55 | 89,29 | 79,36 | 84,03 | 69,62 |
| Glove + GRU | 85,97 | 85,73 | 87,12 | 86,24 | 71,93 |
| **Glove + biGRU** | **86,2** | **86,93** | **85,99** | **86,46** | **72,4** |

*Table 10. Results of Amazon Dataset using transformers*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| Bert + CNN | 92,67 | 89,55 | 97,02 | 93,14 | 85,63 |
| Bert + LSTM | 92,73 | 91,1 | 95,12 | 93,05 | 85,51 |
| Bert + biLSTM | 91,2 | 94,31 | 92,29 | 93,3 | 86,43 |
| **Bert + GRU** | **93,5** | **92,33** | **95,22** | **93,75** | **87,13** |
| Bert + biGRU | 93,37 | 92,56 | 94,69 | 93,61 | 86,76 |
| albert +CNN | 92,68 | 94,43 | 91,07 | 92,72 | 85,41 |
| albert + LSTM | 93,35 | 92,88 | 94,24 | 93,56 | 86,7 |

| | | | | | |
|---|---|---|---|---|---|
| albert + biLSTM | 92,43 | 89,36 | 96,73 | 92,9 | 85,12 |
| albert + GRU | 93,08 | 90,72 | 96,34 | 93,44 | 86,29 |
| **albert + biGRU** | **93,55** | **94,05** | **93,31** | **93,68** | **87,1** |

*Table 11. Results of Yelp Dataset using context-independent embedding*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| FastText + CNN | 91,83 | 90,66 | 93,28 | 91,95 | 83,7 |
| FastText + LSTM | 93,59 | 94,04 | 93,08 | 93,56 | 87,19 |
| FastText + biLSTM | 93,28 | 91,46 | 95,48 | 93,43 | 86,65 |
| **FastText + GRU** | **93,77** | **93,54** | **94,05** | **93,79** | **87,55** |
| FastText + biGRU | 93,37 | 95,12 | 91,43 | 93,24 | 86,8 |
| word2vec + CNN | 92,3 | 92,91 | 91,59 | 92,25 | 84,61 |
| word2vec + LSTM | 94,13 | 93,37 | 95,02 | 94,19 | 88,28 |
| **word2vec + biLSTM** | **94,27** | **95,07** | **93,38** | **94,22** | **88,55** |
| word2vec + GRU | 94,22 | 95,11 | 93,24 | 94,17 | 88,46 |
| word2vec + biGRU | 93,97 | 92,84 | 95,29 | 94,05 | 87,98 |
| Glove + CNN | 91 | 91,56 | 90,32 | 90,94 | 82 |
| Glove + LSTM | 93,43 | 94,5 | 92,22 | 93,35 | 86,88 |
| **Glove + biLSTM** | **93,49** | **93,66** | **93,31** | **93,48** | **86,99** |
| Glove + GRU | 93,2 | 91,61 | 95,11 | 93,33 | 86,46 |
| Glove + biGRU | 92,94 | 91,01 | 95,31 | 93,11 | 85,99 |

*Table 12. Results of Yelp Dataset using transformers*

| Model | Accuracy | Precision | Recall | F1 | MCC |
|---|---|---|---|---|---|
| Bert + CNN | 94,06 | 93,11 | 95,18 | 94,13 | 88,15 |
| Bert + LSTM | 94,48 | 95,68 | 93,17 | 94,41 | 88,99 |
| Bert + biLSTM | 94,58 | 94,55 | 94,62 | 94,58 | 89,16 |
| **Bert + GRU** | **94,64** | **94,88** | **94,38** | **94,63** | **89,28** |
| Bert + biGRU | 94,59 | 95,26 | 93,86 | 94,55 | 89,2 |
| albert +CNN | 93,66 | 96,22 | 90,89 | 93,48 | 87,45 |
| albert + LSTM | 94,13 | 93,63 | 94,7 | 94,16 | 88,26 |
| albert + biLSTM | 94,17 | 93,99 | 94,38 | 94,18 | 88,35 |
| **albert + GRU** | **94,34** | **95,08** | **93,51** | **94,29** | **88,68** |
| albert + biGRU | 94,33 | 93,98 | 94,73 | 94,35 | 88,66 |