applied sciences

Article

An Emotion-Based Rating System for Books Using Sentiment Analysis and Machine Learning in the Cloud

Sandhya Devi Gogula [1], Mohamed Rahouti [2,*], Suvarna Kumar Gogula [3], Anitha Jalamuri [4] and Senthil Kumar Jagatheesaperumal [5]

[1] Department of Computer Science Engineering, GITAM School of Technology, Visakhapatnam 530045, India
[2] Department of Computer & Information Science, Fordham University, Bronx, NY 10458, USA
[3] Department of Computer Science Engineering, MVGR College of Engineering, Vizianagaram 535005, India
[4] Department of Computer Science Engineering, Malla Reddy Engineering College, Hyderabad 500100, India
[5] Department of Electronics & Communication Engineering, Mepco Schlenk Engineering College, Sivakasi 626005, India
* Correspondence: mrahouti@fordham.edu

Abstract: Sentiment analysis (SA), and emotion detection and recognition from text (EDRT) are recent areas of study that are closely related to each other. Sentiment analysis strives to identify and detect neutral, positive, or negative feelings from text. On the other hand, emotion analysis seeks to identify and distinguish types of feelings such as happiness, surprise, grief, disgust, fear, and anger through the expression of texts. We suggest a four-level strategy in this paper for recommending the best book to users. The levels include semantic network grouping of comparable sentences, sentiment analysis, reviewer clustering, and recommendation system. The semantic network groups comparable sentences at the first level utilizing pre-processed data from reviewer and book datasets using the parts of speech (POS) tagger. In order to extract keywords from the pre-processed data, feature extraction uses the bag of words (BOW) and term frequency-inverse document frequency (TF-IDF) approaches. SA is performed at the second level in two phases: training and testing, employing deep learning methodologies such as convolutional neural networks (CNN)-long short-term memory (LSTM). The results of this level are sent into the third level (clustering), which uses the clustering method to group the reviewers by age, location, and gender. In the last level, the model assessment is carried out with accuracy, precision, recall, sensitivity, specificity, G-mean, and F1-measure. The book suggestion system is designed to provide the highest level of accuracy within a minimum number of epochs when compared to the state-of-the methods, SVM, CNN, ANN, LSTM, and Bi-directional (BI)-LSTM.

Keywords: books recommendation; sentiment analysis; machine learning; cloud; CNN; LSTM

check for updates

Citation: Gogula, S.D.; Rahouti, M.; Gogula, S.K.; Jalamuri, A.; Jagatheesaperumal, S.K. An Emotion-Based Rating System for Books Using Sentiment Analysis and Machine Learning in the Cloud. *Appl. Sci.* **2023**, *13*, 773. https://doi.org/10.3390/app13020773

Academic Editors: Huan Wang and Wen Zhang

Received: 9 December 2022
Revised: 29 December 2022
Accepted: 30 December 2022
Published: 5 January 2023

1. Introduction

Sentimental analysis is one of the popular machine learning approaches that support the identification of feelings. It allows businesses to collect valuable information about their customers' preferences through various social media platforms, surveys, and e-commerce website evaluations [1]. Sentiment analysis drives the study of analyzing client views, expressions, preferences, and dislikes toward various things such as products, services, companies, and persons. People may express their experiences regarding various items through reviews, comments, or ratings on social media and e-commerce platforms such as Amazon, Flipkart, and others [2]. The reputation of a product is determined by the collective opinion of its internet users. Sentiment analysis, also known as computational analysis of opinion, has received much interest due to its future uses in e-commerce, review sites, and online discussion forums. Sentiment analysis is often challenging since it needs to provide better traditional lexical categorization [3]. This happens because the evaluations are unstructured and written in plain English. Additionally, evaluations can be published in non-grammatically exact local languages and slang. The historical parsers created previously may not be suited for such material.

For example, imagine someone wrote on CookingLight.com (Recipe Site) about a recipe for Mexican Pizza and said, “Though I attempted the same method as stated, the meal tasted excessively salty”. We can use sentiment analysis to see whether the reviewer enjoyed the food. “Too salty” can be interpreted as a negative attribute, indicating that the client did not want the food [4].

Sentiment analysis uses novel categorization algorithms to solve these issues. Its purpose might be as primary as a positive or negative (binary classification) or as complex as a multi-class classification. It is often recommended to customize opinion mining to extract subject-specific evaluations because reviews contain people’s thoughts on several issues. Sentiment analysis can also be conducted on a broader or more detailed level [5]. The method of identifying whether any piece of literature, an abstract, or a sentence is neutral, positive, or negative is often categorized as a sentiment analysis. It is also referred to as opinion mining since it derives a speaker’s or writer’s viewpoint on a topic or the general context of a document [6]. For online reviews or social media, sentiment analysis is employed to ascertain how customers or people feel about a problem. Finding the text’s polarity in a sentence or document, or whether their expressed attitude is neutral, positive, or negative, is one of the main objectives of sentiment analysis. Figure 1 shows the stages involved in a sample sentiment analysis in a product review.

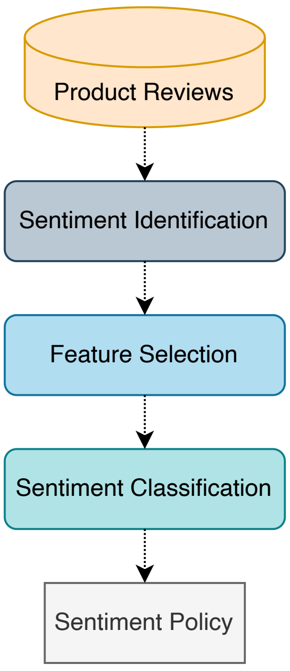


Figure 1. A sample sentiment analysis stages involved in a product review.

People today produce a massive volume of information through casual writing in the Internet age. Most social networking platforms offer challenges such as improper grammar, spelling mistakes, and strange vocabulary. These difficulties make sentiment and emotion analysis challenging for robots [7]. People do not always clearly express their emotions. For instance, when asking “Why have you been so late?”, “why” might be spelled incorrectly as “y”, the word “you” might also be misspelled as “u”, and “soooo” might be use to emphasize a point.

The authors in [8] described the design phases of an ML-based classifier for detecting the failure rates in assessing a trained recurrent deep neural network (RNN). The work focused on assessing the defective water meters is highly applicable to the context of adopting the RNN framework for accessing the semantics involved in the book recommendation system. Furthermore, the work in [9] addresses the means to handle the imbalanced data available in a corpus of big data. It examines the defects using deep learning frameworks and provides insights into the strengths and weakness of each approach to provide an effective means of handling imbalanced data, which is largely assistive for book recommendation strategies. Furthermore, the wording needs to clarify whether the speaker is angry or worried. As a result, it can be challenging to infer moods and emotions from real-world text data for several different reasons. Lack of resources is also one of the difficulties encountered in emotion detection and sentiment analysis. For instance, a large dataset that has been annotated is required for several statistical processes. Data collection is simple, but manually classifying a large dataset takes a lot of effort and needs to be more accurate. The fact that the vast bulk of the materials is only available in English is another problem. As a result, sentiment analysis and emotion detection from non-English languages, particularly regional languages, present both a significant difficulty and a fantastic opportunity for academics [10]. Additionally, some lexicons and corpora are domain-specific, which limits their applicability to other disciplines.

The abnormal slags used in social media platforms, termed Web slang, are another issue frequent in social media chats and posts on Instagram, Facebook, and Twitter. For instance, most of the younger generation utilizes terms such as ‘LOL’, which stands for ‘laughing out loud’, and ‘FOMO’, which stands for ‘fear of missing out’ and denotes worry [11]. Existing lexicons and trained models face a significant challenge as the vocabulary of Web slang grows. Sarcastic and ironic words are commonly used to communicate wrath or disappointment, which might be challenging to discern. For example, the adjective excellent in the line “This narrative is great to put you to sleep” denotes a favorable attitude, although the reviewer found it to be highly uninteresting. As a result, detecting sarcasm has become a time-consuming problem in the field of mood and emotion recognition [12].

The way to express numerous emotions through a single statement is another issue. A multi-opinionated statement makes it difficult to discern numerous features and their accompanying attitudes or emotions. For example, the line “the view from this point is so tranquil and quiet, but this area smells” demonstrates two emotions: revulsion and calm. The inability to clearly distinguish polarity while comparing phrases is another problem. Consider the statements “Phone A is worse than Phone B” and “Phone B is worse than Phone” [13]. Although the word “worse” conveys negative polarity in both words, they could not be further apart.

2. Methodology and Methods Framework

As illustrated in Figure 2, the sentiment analysis and emotion detection method involve several steps, including dataset collection, pre-processing, feature extraction, model construction, and assessment.

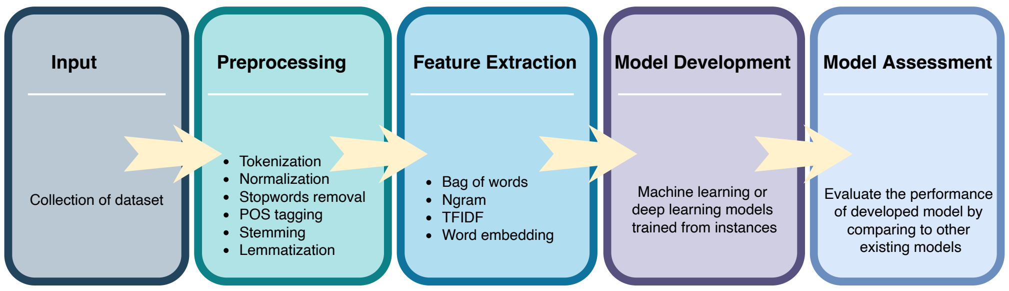


Figure 2. Block diagram for the proposed framework.

2.1. Data Collection

Researchers have employed a variety of mood and emotion analysis datasets to measure the performance of their models, as shown in Table 1 and Figure 3. The datasets most frequently used in sentiment and emotion analysis are Stanford Sentiment Treebank

(SST), SemEval, and the International Survey of Emotional Antecedents and Reactions (ISEAR). The SemEval and SST datasets come in various flavors, each with its domain, size, and other characteristics. ISEAR was created from a group of people who experienced all seven emotions (included in Table 1) in various situations. As shown in the table, it lists tales, reviews, comments, and tweets as the most popular datasets. The EmoBank dataset, which was created from news, blogs, and letters, was subjected to the Valence, Arousal Dominance Model (VAD) dimensional analysis. Information from social media platforms such as Facebook, YouTube, and Twitter has been collected by numerous studies and classified in the literature by experts in language and psychology [14]. Postings, blogs, and e-commerce sites on various social media platforms generally contain unstructured data. It must be categorized to avoid additional calculations described in the next section.

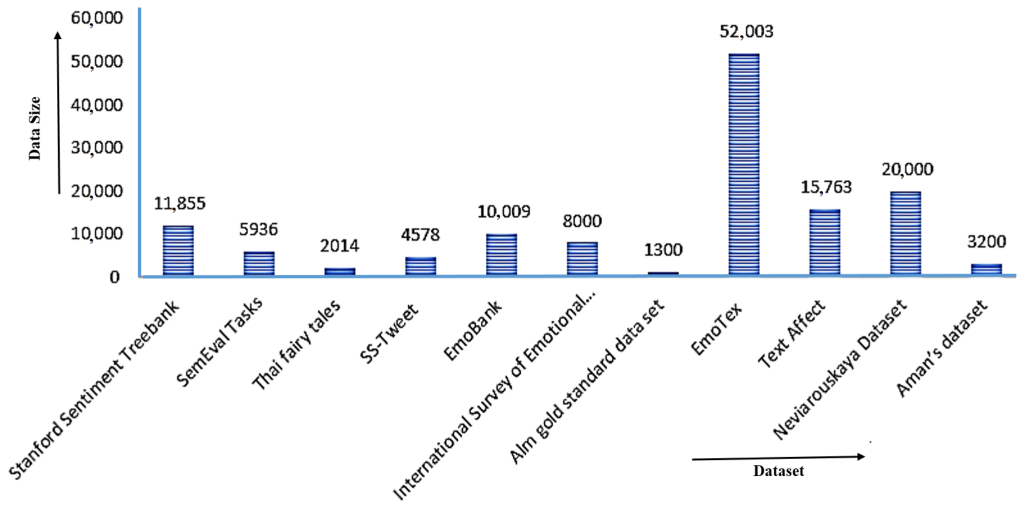


Figure 3. Emotion detection datasets with their corresponding data sizes. ("International Survey of Emotional ..." means "International Survey of Emotional Antecedents").

Table 1. Sentiment analysis and emotion detection datasets.

Dataset	Data Size	Sentiments/Emotions	Range
Stanford Sentiment Treebank	11,855	Excellent, excellent, very negative, very negative, and neutral.	5
SemEval Tasks	5936	neutral, adverse, and positive	3
Thai fairy tales	2014	negative, neutral, and positive	3
SS-Tweet	4578	Strengths both positive and negative	1 to 5 for positive and −1 to −5 for negative
EmoBank	10,009	Neutral, excellent, and very negative.	5
International Survey of Emotional Antecedents and Reactions	8000	Neutral, extremely positive, very positive, negative, and extremely negative.	7

Table 1. *Cont.*

Dataset	Data Size	Sentiments/Emotions	Range
Alm gold standard data set	1300	Feelings of joy, fear, sadness, surprise, and disgust-anger	5
EmoTex	52,003	Excellent, excellent, very negative, very negative, and neutral.	5
Text Affect	15,763	Extremely positive, very positive, negative, extremely negative, and neutral.	6
Neviarouskaya Dataset	20,000	Very positive, very positive, very negative, and neutral.	10
Aman's dataset	3200	Neutral, Positive, very positive, negative, and very negative.	7

2.2. Pre-Processing of Text

People constantly express their views and feelings in a sincere way on various social media platforms. Since the information extracted from such social media platform comments, audits, posts, remarks, and complaints are often unstructured, performing sentiment and emotion analysis for computers is challenging [15]. Subsequently, pre-processing is an essential phase in the data-cleaning process because it significantly influences many subsequent operations. Pre-processing techniques such as stop word removal, tokenization, and POS labeling are required due to the nature of a dataset. Action must be taken since some of these pre-processing methods could cause data loss essential for sentiment and emotion research. Two crucial stages in pre-processing are lemmatization and stemming. Truncating suffixes are used in stemming when restoring words to their original form [16]. For instance, "argued" and "argue" are changed to "argue". With this approach, there is no longer a need for pointless sentence calculations. The lemmatization process consists of removing inflectional ends from a token to transform it into the lemma at its base. For example, the word "caught" is changed to "catch". On two datasets, SS-Tweet and SemEval, researchers evaluated using combination and ablation analysis on the performance of four machine learning models through several pre-processing strategies. Lemmatization and number deletion had no effect on accuracy, but punctuation deletion did, according to the scientists' findings.

To determine the grammatical category to which a word belongs based on context, such as whether it is an adjective, adverb, noun, verb, or another part-of-speech tagging (POS tagging) is used, following a relationship analysis of the phrase, each word is assigned the proper tag [17]. Text normalization attempts to make the text less unpredictable and to bring it closer to a predetermined standard. Giving succeeding algorithms less unique information to work with increases efficiency using two methods, stemming and lemmatization.

Such normalization methods seek to reduce a word's inflectional and occasionally derivative forms to their basic form. The process of stemming involves reducing words to their basic form. Using a list of frequently occurring prefixes and suffixes, stemming algorithms work by deleting a word's starting or ending [18]. This haphazard cutting does not always succeed. Thus, this tactic has many drawbacks, and the words could be reduced to their purest form through lemmatization. The lemmatization algorithms correctly reduce inflected words to preserve the linguistic relationship to the underlying word. The stemming algorithm may not be able to distinguish between words with multiple meanings based on the word type since it only understands the context of the individual word rather than the context of the entire phrase. This is the distinction between lemmatization and stemming. Stemming algorithms have the benefit of being simpler to design and apply. Stemming algorithms are the best choice if precision is not required for the application [19]. Lemmatization algorithms take longer to operate because each component of speech for a word must be determined at the beginning, and the normalization rules for each part of speech must be calculated independently. The Porter Stemmer is possibly the most well-known stemming technique (we will also use it for this task). However, there are various alternative options, such as Lovin's, Dawson's, Krovetz, Xerox, and Snowball Stemmer. There are also various algorithms for lemmatization, just as for stemming, such as spaCy, TextBlob, Stanford CoreNLP, and Gensim Lemmatize.

For the following examples, we make use of the NLTK Word Net Lemmatizer. A particular POS tag can be defined using the Wordnet Lemmatizer. This setting should be pos = 'v', where 'v' stands for "verb". This POS tag is used frequently. "Are" has been changed to "be", and "eating" and "swimming", respectively, have been changed to "eat" and "swim". The word "saw" is an exception. One would think that this phrase would be replaced with "see". Consider the sentence, "I want a better dog". For "better", we like to use "good" or "okay". This is an adjective rather than a verb; thus, we must apply the Word Net Lemmatizer with the POS tag = "a" for adjective [20]. Additionally, we developed a function that achieves this. However, the verbs were not taken into account. One choice to address this is by building a function that extracts a word's POS tag and passes it to the token's lemmatization function. A few of the sample part-of-speech constants are ADJ = 'a', ADJ_SAT = 's', ADV = 'r', NOUN = 'n' and VERB = 'v'.

2.3. Feature Extraction

The computer interprets text using numbers. Word vectorization, also known as word embedding, converts words or text into vectors of real value. This feature extraction strategy divides a text into sentences, which are subsequently broken down into texts, and creates a matrix or a feature map as a result [21]. Every individual feature column in the final obtained matrix denotes a document or phrase, and the values present in the feature map's cells typically indicate the frequency of the word in the document or phrase. "Bag of Terms" (BOW), one of the most straightforward techniques for feature extraction, defines a fixed-length vector of the count, where every item corresponds to terms in a specified dictionary. If a word in a sentence is absent from the pre-defined dictionary, it obtains a score of 0; otherwise, depending on the frequency of its appearance in the sentence, it is assigned a value greater than or equal to 1. As a result, the vector's length is always equal to the entire set of words in the English language. The simplicity of this approach offers advantages, but there are also several disadvantages, including the generation of a sparse matrix, the loss of the order of words in a phrase, and the inability to accurately capture the content of a sentence. For instance, using the pre-defined existing dictionary, the line "are you enjoying reading" might be represented as we, hope, you, are, enjoying, reading. However, text pre-processing, n-grams, and TF-IDF can be used to enhance these representations.

In phrase vector encoding, the N-gram approach serves as a good stand-in for word order resolution. Here, the words are represented as a combination of discrete n-gram meaning groupings of n number of adjacent keywords or words in an n-gram vector representation. In this context, any of the natural numbers can be used as n. The trigrams such as "to teach is", "is to touch", "teach is to", "is to touch a life", and "a life forever" is formed from the phrase "to teach is to touch a life forever". The sentences' original arrangement is maintained as a result. N-gram features outperform the BOW method because they contain grammatical patterns that convey essential data. Despite maintaining word order, n-grams lack data and are highly dimensional [22].

The term frequency-inverse document frequency (TF-IDF) technique is one of the well-liked feature extraction strategies. This technique encodes text as a matrix, where each integer corresponds to the amount of data each phrase in a document contains [23]. It is based on the notion that uncommon words in texts contain rich knowledge. IDF is computed from the ratio of the total number of documents (N) to the total number of documents in which a word W appears n. Term frequency is computed from the ratio of the total number of words W in a document by the number of times a word W is present in that document. To choose the most effective methodology, we tested two feature extraction methodologies and used six pre-processing procedures. They employed six machine learning techniques, including n-grams with $n = 2$ and TF-IDF, to extract characteristics from the dataset of SS-tweet and found that TF-IDF outperformed n-gram [24]. A deep learning network can choose the best vector representations because of the large amount of data. Using neural network-based word embedding to extract features results in more informative features. Words with similar meanings or those connected to one another are represented by comparable vector embedding in word representation based on neural networks [25].

Due to the preservation of word semantics, this is more frequent in word prediction. Researchers at Stanford University created GloVe, a deep learning-based word embedding technique, and Facebook introduced FastText. Word2vec vectors take longer to train than GloVe vectors and cannot compete with FastText vectors in several ways. Furthermore, it was shown that even when utilizing phrases that are out of vocabulary (OOV), using neural networks to select acceptable word embeddings may lead to significant benefits. Several word embeddings that were trained using Twitter and Wikipedia as corpora were compared using TF-IDF word embedding [26]. The abbreviation for frequency is TF-IDF. Inverse document frequency records involves figuring out how pertinent a word present in a corpus or series is turned out to be a text. A word's meaning deepens in direct proportion to how frequently it appears in the text, but the corpus's word frequency cancels this out. A sample pre-processing output of text from documents is shown in Table 2.

Term Frequency (TF): The frequency in document d indicates how frequently t a specific word appears. Subsequently, a term becomes more meaningful the more times it is present in the text, which makes sense. Due to the irrelevance in the sequence of the terms, a vector can be used to validate the text present in the bag of word models. Here, each phrase in the document has its own entry, with the word frequency serving as the entry's value. Simply put, a term's weight in a document is inversely correlated with its frequency.

$$tf(t,d) = \frac{count\ of\ t\ in\ d}{number\ of\ words\ in\ d} \tag{1}$$

Document Frequency (DF): It verifies the text's meaning throughout the entire corpus collection, which is very comparable to TF. Here, the only distinction in document d, tf denotes a term's frequency counter, whereas df denotes the term's frequency in document set N. That is, df represents the number of publications in the term.

$$df(t,d) = occurrence\ of\ t\ in\ documents \tag{2}$$

Inverse Document Frequency (IDF): Primarily, it evaluates the word's applicability. Finding records that match the criteria is the search's primary objective. The term frequencies cannot be used to gauge a term's importance in the document because tf views all words as having similar meanings. To begin, determine the phrase's document frequency by counting the number of papers that is present in the phrase "t":

$$df(t,d) = N(t) \tag{3}$$

where $df(t)$ refers to the document frequency of a specific term t, and $N(t)$ denotes the number of documents that include the term t.

Contrary to the frequency of text present in a document, which is reliant on the whole corpus and represents the significant number of independent documents where a word appears, phrase frequency is defined as the number of times it appears in a single document. By visualizing how the frequency of the inverse text is specified, obtained through the ratio of the number of documents in the corpus to the frequency of the text, subsequently, the IDF of the phrase is determined.

$$idf(t,d) = \frac{N}{df(t)} = \frac{N}{N(t)} \tag{4}$$

Table 2. Sample pre-processing output.

Name	Review	Country	Rating	Date
C. Whitmore	"Between Two Kingdoms" was checked out from my local library, and I was eager to read about another young adult survivor's journey. Due to my own thirteen-year survivor status from a different, less demanding disease, I am an ardent reader of this specific genre. My personal quest for meaning continues, and I read works like this in the hopes of gaining some fresh understanding.	United States	3	4 March 2021
Kate HK	Suleika's narrative drew me in right away and kept me there till the very end. Her account of living in the realm of disease and then navigating her way back to the kingdom of wellness is fascinating. I was obliged to accompany her on this voyage, to go through these landscapes with her, and to discover where she settles. What happens to her in the story is intriguing, but what occurs within her is the actual story.	United States	5	13 February 2021
Rushmore	I thought I knew what to anticipate, but this book exceeded my expectations. I was familiar with Suleika's narrative and writing since I participated in her Isolation Journals project (via the regular journaling prompts she provides). I've attended a dozen teleconferences led by Suleika in the last year. She is always gracious, pleasant, and attentive. I never felt pressured or frightened into buying the book since she provided bits and pieces of the revising process and her excitement for its imminent release. I bought it as a token of my appreciation, not knowing when I'd get around to reading it.	Canada	5	15 February 2021
Amy T	The writer describes her cancer diagnosis, treatment, and interactions with her partner, family, and other cancer sufferers with great care. Her recovery from near death was extraordinary. Her personal progress from being completely reliant on others to be self-sufficient with the capacity and motivation to do a solo road trip was the most significant shift.	U.K	4	2 June 2021
Jane	The author's ability to overcome obstacles astounded me. Most people would be stopped in their tracks by this disease, but it seemed to drive her to keep going until she reached the other side. Maybe it was her youth, but no one could have gotten through this without her incredible fortitude. It will put all of your concerns and criticisms into context.	India	4	5 April 2021
Ellie Rhodes	I bought this book with no understanding of what it was about and then realized I didn't care. I would have preferred to learn more about the trip and less about the minor details. It might have benefited from further editing.	Brazil	2	24 May 2021

A more popular word is intended to be viewed as less important, even though the element (most definite integers) looks unduly severe. The inverse frequency of the paper is then determined as its logarithm (with base 2). As a result, the term t is as follows:

$$idf(t,d) = log\frac{N}{df(t)} \tag{5}$$

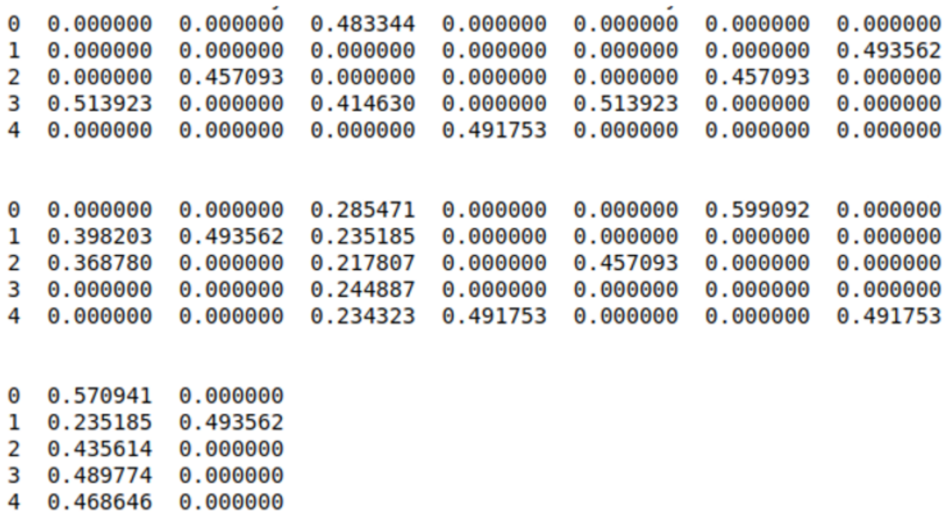
Computation: One of the most influential metrics for assessing a phrase's significance within a collection or corpus of texts is tf-idf. The tf-idf scheme of weighting gives every individual word in a document a corresponding weight based on its term frequency tf and a reciprocal document frequency termed as $tf - idf$. More meaningful words tend to have higher weight ratings. The tf-idf weight is usually made up of normalized term frequency ntf and inverse document frequency idf.

$$tf - idf(t,d) = tf(t,d) \times idf(t) \tag{6}$$

Phrase frequency analysis examines how frequently a term appears compared to other terms in the document. There are various techniques for figuring out frequency:

- How frequently a term is used in a text (raw count);
- Due to the length of the content, term frequency has been modified;
- Frequency need to be scaled up logarithmically such that $log(1 + rawcount)$;
- Regularize the boolean expression with logic 1 and 0 representing the presence and absence of the terms, respectively;
- The inverse document frequency visualizes the term's rarity or frequency within the corpus. This is how the IDF is calculated: N is the number of documents d in the corpus, and t is the term (word) whose frequency we want to gauge D. The only information the numerator presents is the number of publications that use the word "t".

  Figure 4 shows the feature extraction output.



```
0  0.000000  0.000000  0.483344  0.000000  0.000000  0.000000  0.000000
1  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.493562
2  0.000000  0.457093  0.000000  0.000000  0.000000  0.457093  0.000000
3  0.513923  0.000000  0.414630  0.000000  0.513923  0.000000  0.000000
4  0.000000  0.000000  0.000000  0.491753  0.000000  0.000000  0.000000


0  0.000000  0.000000  0.285471  0.000000  0.000000  0.599092  0.000000
1  0.398203  0.493562  0.235185  0.000000  0.000000  0.000000  0.000000
2  0.368780  0.000000  0.217807  0.000000  0.457093  0.000000  0.000000
3  0.000000  0.000000  0.244887  0.000000  0.000000  0.000000  0.000000
4  0.000000  0.000000  0.234323  0.491753  0.000000  0.000000  0.491753


0  0.570941  0.000000
1  0.235185  0.493562
2  0.435614  0.000000
3  0.489774  0.000000
4  0.468646  0.000000
```

Figure 4. Features' extracted values.

$$idf(t,d) = log(\frac{N}{count(d \in D : t \in d)}) \tag{7}$$

Scikit–learn

$$IDF(t) = log(\frac{1+n}{1+df(t)}) + 1 \tag{8}$$

Standard notation

$$IDF(t) = log(\frac{n}{df(t)}) + 1 \tag{9}$$

Moreover, IDF is needed to assist in correcting words such as "as", "of", and "the", which are frequently found in an English corpus. We can decrease the weighting of popular terms while raising the significance of uncommon phrases by applying inverse document frequency. Finally, IDFs can be obtained from the dataset used in the current experiment and a corpus present in the background that corrects the sample bias.

TF-IDF: The core tenet of TF-IDF is that the frequency of a term across texts is inversely associated with its significance. The IDF function indicates how uncommon a term is in the collection of documents, whereas the TF function indicates how frequently a phrase appears in a document. Our final TF-IDF value can be obtained by summing these numbers.

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{10}$$

The more important or relevant a term is, the higher its TF-IDF score; the less significant or relevant a phrase is, the lower its TF-IDF value.

3. Model Development

Three 1D convolutional layers make up the feature extraction block in the convolutional neural networks (CNN), and between the two convolution layers, we used the MaxPooling and the Rectified Linear Unit (ReLU) layers. Convolutional layers enable early layers in a deep learning architecture to learn the properties of low-level obtained from the applied input. The output of the convolutional layers, the feature map, can only track the precise locations of the input features. It implies that even little changes in the input feature's position will result in various feature maps [27]. Figure 5 shows the sequence of stages involved in the proposed deep learning framework developed using CNN-LSTM architecture.

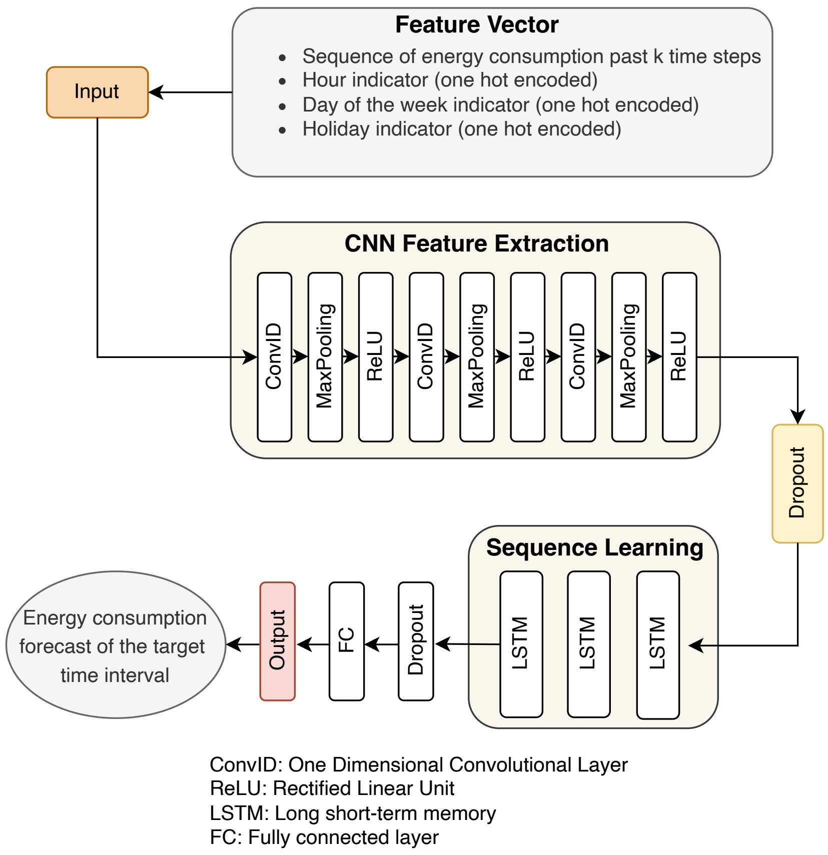

Feature Vector
Sequence of energy consumption past k time steps
Hour indicator (one hot encoded)
Day of the week indicator (one hot encoded)
Holiday indicator (one hot encoded)
Input
CNN Feature Extraction
ConvlD
MaxPooling
ReLU
ConvlD
MaxPooling
ReLU
ConvlD
MaxPooling
ReLU
Dropout
Sequence Learning
LSTM
LSTM
LSTM
Dropout
FC
Output
Energy consumption forecast of the target time interval
ConvlD: One Dimensional Convolutional Layer
ReLU: Rectified Linear Unit
LSTM: Long short-term memory
FC: Fully connected layer

Figure 5. The proposed architecture developed using CNN-LSTM-based deep learning frameworks.

Typically, a pooling layer is included after the convolutional layer to reduce the limitation in the invariance of the resulting feature map. The activation function is employed to improve the model's capacity to learn complex structures. Furthermore, we have added a MaxPooling layer to our model to lessen the overall computational burden. With the downsampling method MaxPooling, feature maps' spatial dimensions are cut in half. Because the Rectified Linear Unit (ReLU) activation function is immune to the gradient vanishing problem, researchers typically use it to improve the network's trainability [28].

A great way to avoid overfitting while developing the deep learning model is to use the dropout layer. During training, some neurons in this layer are silenced after being randomly selected. In this study, we added a dropout layer to prevent overfitting between the sequence learning of LSTM and the feature extraction block of CNN. The final output is formed by connecting the sequence learning block output to a dropout layer, which is then connected to a fully connected layer [29]. When creating a CNN model, a coarse-to-fine technique is commonly employed. The vast number of trainable parameters in this structure increases its computational complexity. For our CNN feature extraction block, we chose a pyramidal structure where the number of kernels is massive at the base and steadily decreased as we go down to the lower levels. The first convolution layer's kernel size is 48, while the sizes for the next two convolution layers are 32 and 16, respectively. This structure avoids overfitting by lowering the number of parameters that can be trained. Three LSTM layers present in the sequence learning block, each with 20 neurons, were used. The return sequence for the first two LSTM layers is set to true to allow the network to present the output to the full series of hidden states. The return sequence for the last LSTM layer is set to false to enable the hidden state as an output of the network at the final time step. The dropout layer was employed prior to the fully linked layer to avoid over-fitting. There are 20 neurons in the fully connected layer. To test different lookahead counts, the output layers' number of neurons is increased from one to six. The parameters for the built-in deep learning system are presented in Table 3. Using "Adam" as one of the well-known optimizers, in this study, we employed mean absolute error (MAE) as a loss function.

Table 3. Settings of the parameters chosen in the developed model.

Parameter	Setting
Learning Rate	0.1
Optimizer	Adam
Adjustment	Factor = 0.7, min LE = 1×10^{-4}
Batch Size	256
Epoch	500

The suggested deep learning model's training flow is shown in Figure 6; 10% test, 20% validation, and 70% training data were produced. To track the validation loss, we employed the mean absolute error (MAE) as a loss function. After the training data and validation data are loaded, the training procedure is started. The validation loss is set for each epoch, and its progress is monitored. If the validation loss decreases, the developed model is saved with the modified weights and extended epochs. The learning rate is slowed down, and the total number of epochs is increased if the validation loss does not reduce for ten successive epochs. The training is finished once 150 epochs have passed.

LSTM can be used to optimize recurrent neural networks (RNNs), which are extensively used in prediction and time series analysis. It has a more intricate internal structure and uses a particular gating unit to convey information selectively across a recurrent network structure of neurons [30]. In the evaluation stages, the $'tanh'$ provides the hyperbolic tangent function, where U, W, and b are the parameters for the bias vector, weight matrix, and the Sigmoid activation function.

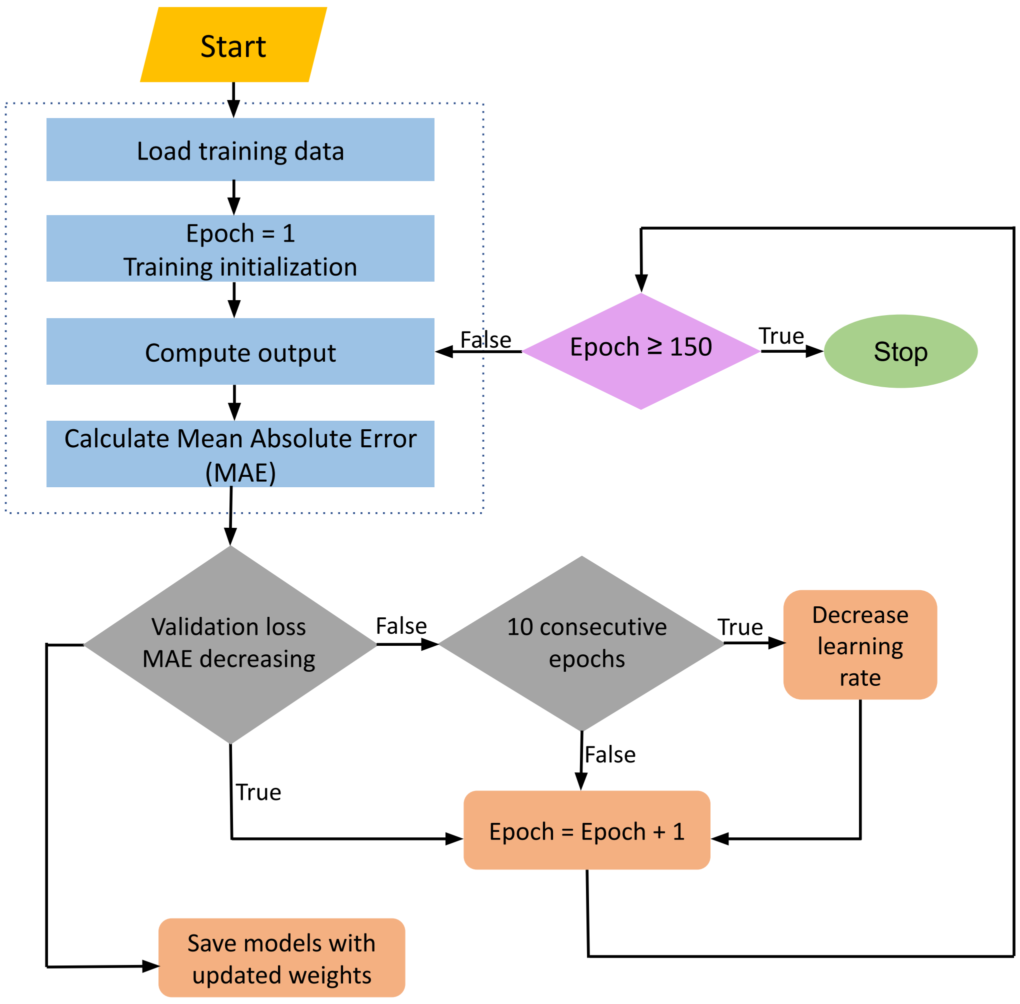


Figure 6. Sequence of stages involved in the training phase of the proposed model.

While the output value is 1, the matching value should be saved; if it is 0, it should be destroyed entirely. The memory gate's *tanh* function is used to decide how much fresh data should be reflected in the state of the cell.

$$ft = \sigma(W_f x_t + U_f h_t - 1 + b_f) \tag{11}$$

$$it = \sigma(W_i + U_i h_t - 1 + b_i) \tag{12}$$

$$(C_t) = tanh(W_c x_t + U_c h_t - 1 + bc) \tag{13}$$

$$C_t = f_t \times C_t - 1 \times i_t \times C \tag{14}$$

$$O_t = \sigma(W_o x_t + U_o h_t - 1 + b_o) \tag{15}$$

$$h_t = O_t \times tanh(C_t) \tag{16}$$

The information to be reflected is then computed by adding the previously computed values through the dot product produced by the forgetting gate, the dot product, and the prior cell state values. The value computed by the output gate is ultimately acquired by multiplying the computed unitary state value by the output value. The output states are then transferred to the attention mechanism, where the attention is incorporated into our model to better comprehend the crucial information present in the recent codes and to acquire key spatiotemporal characteristics. In order to improve the newly encoded attributes, it pays particular attention to specific patterns or the training dataset changes and gives them bigger probability weights. The attention mathematical equation is as follows:

$$e_{ij} = tanh(W_1 h_i + W_2 h_j + b\alpha) \tag{17}$$

$$a_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij}}{\sum_j exp(e_{ij})} \tag{18}$$

$$H_i = \sum_j a_{ij} h_j \tag{19}$$

where does the relationship between values exist? "Weight", "bias", "value of attentional weight for the software function", as well as the "output's final state after passing attention" are all represented by the letter W. Furthermore, we apply the abovementioned method to generate another code with additional features to determine the new code's matching weight.

Moreover, fewer panning operations should be performed to enhance the model. The CNN can be trained with fewer parameters, redundant training data, training weights, and filters using a technique called pooling. Max pooling is one of the most used pooling operations. The fully linked layer produces the final 1-D output data and uses the following algorithm to determine the score:

$$y_j = \sum_{i=1}^{N} w_{i,j} x_i + b \tag{20}$$

where the relationship between the input value and the neuron weight is x, the input data length is N, and the output of the entire linked layer is y. Furthermore, b represents the bias and denotes the neuron [31]. For higher-level linked units to decide how much to contribute to the ensuing forecast, the fully connected layer uses an activation function to send the values of the outcomes to those units. The activation function is represented as follows:

$$\mu_j = g(y_j) = max(0, y_j) \tag{21}$$

The outcome of the activation function is Equation (21). We effectively avoid overfitting by using ReLU as the activation function. The dataset is run through a different activation function termed the linear activation function "n" before the final result is produced.

CNN: Text processing is one area where CNN, an artificial neural network, excels. Here, sharing the weights of the convolution operation is crucial because all text analytics uses the kernel. Weight sharing allows the kernel, through downsampling, to learn about the feature patterns and their spatial hierarchy, along with the pooling operations for capturing a significantly larger field of view, in contrast to a completely connected network.

$$\mu_j = f(y_j) = \mu_j \tag{22}$$

The pre-processed book review data are sent to the LSTM model, which then generates numerous new sequences with input and output book review sequences of various lengths [32]. Recent data sequences are fed as input into the attention mechanism to improve temporal properties. Furthermore, the CNN model gathers the output spatial features from the text, decreases the parameters, and uses the maximum pooling layer to achieve the redundancy function. Last, the information is sent to the final hidden layer through a layer linear activation function and the fully connected layer, resulting in the final prediction.

CNN-LSTM Model: This study suggests using the RAdam optimization algorithm to improve the convergence and robustness of the prediction model developed using CNN-LSTM while increasing the spatiotemporal feature extraction from text sequences. This significantly reduces the error in the prediction process of the developed model by shortening the training time of the model and ultimately results in better perfection in prediction accuracy, which turns out to be a part of the model-based methodology created for this study.

Figure 7 depicts the stages flow in the proposed CNN-LSTM model developed in this work. Outlier removal, linear interpolation, and data normalization processing are examples of data preparation. In the experimentation carried out in this study, the enhanced CNN-LSTM model is applied using the following stages:

1. The first iteration of the text spatiotemporal correlation feature matrix is built using the historical text data.

$$\begin{bmatrix} x_1^1 & \cdots & x_{t-m}^1 \\ \vdots & \ddots & \vdots \\ x_1^i & \cdots & x_{t-m}^1 \end{bmatrix} \tag{23}$$

   Where was the historical data taken from, and when were the book review frames collected?
2. The multilayer CNN produces the text sequence feature vector by extracting the spatial properties of the text data from the feature matrix.
3. By means of feeding the multilayer LSTM network with the feature vectors, certain time-dependent features may be extracted.
4. After the completion of the model network structure, the dropout layer is added, and the training procedure is then optimized using the RAdam method to avoid data overfitting and protracted training convergence periods. An algorithm is used to refer to the RAdam optimization algorithm.
5. Following model training and testing, the following is the output of the anticipated text sequence:

$$h = \left\{ h_{t-(m-1)}, h_{t-(m-2)}, h_{t-(m-3)} \ldots, h_t \right\} \tag{24}$$

   where h is the expected text sequence at time t, which is the estimated text at time t.

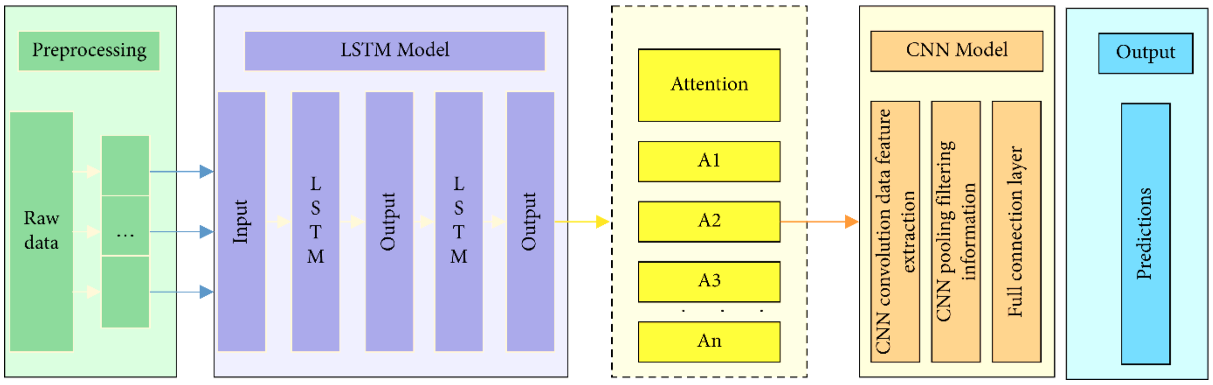


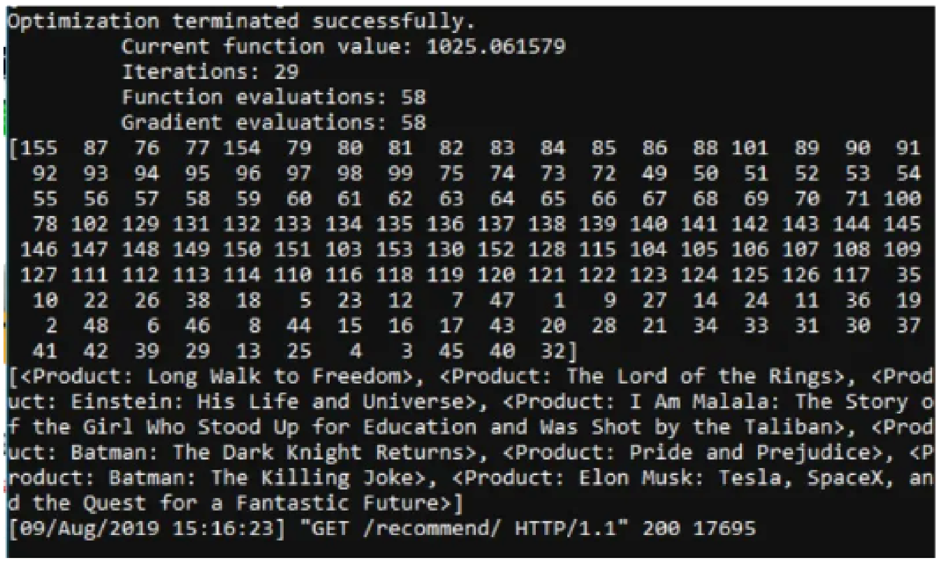
Figure 7. Fused attention mechanism based representation of flow of stages in the proposed CNN-LSTM model.

4. Results and Discussion

The model is then compared against baseline models using several factors. To evaluate the performance of the models, model assessment measures are required [33]. The number of predictions, which were categorized as correct and incorrect, are made using real numbers that are known and can be determined using a confusion matrix. For imparting data fitting based on classes of positive and negative, the confusion matrix shows True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) values. In addition to the F1 score, the model was tested using recall, accuracy, and precision metrics. Figure 8 shows the CNN-LSTM model output obtained after implementation.

Accuracy: It measures how well the model is capable of performing better across all types of classes. It is particularly estimated when all decisions are equally important and helpful. It is determined by dividing the proportion of accurate judgments by the total volume of judgments made.

$$Accuracy = ((TP + TN))/((TP + TN + FP + FN)) \tag{25}$$



```
Optimization terminated successfully.
         Current function value: 1025.061579
         Iterations: 29
         Function evaluations: 58
         Gradient evaluations: 58
[155  87  76  77 154  79  80  81  82  83  84  85  86  88 101  89  90  91
  92  93  94  95  96  97  98  99  75  74  73  72  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 100
  78 102 129 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
 146 147 148 149 150 151 103 153 130 152 128 115 104 105 106 107 108 109
 127 111 112 113 114 110 116 118 119 120 121 122 123 124 125 126 117  35
  10  22  26  38  18   5  23  12   7  47   1   9  27  14  24  11  36  19
   2  48   6  46   8  44  15  16  17  43  20  28  21  34  33  31  30  37
  41  42  39  29  13  25   4   3  45  40  32]
[<Product: Long Walk to Freedom>, <Product: The Lord of the Rings>, <Prod
uct: Einstein: His Life and Universe>, <Product: I Am Malala: The Story o
f the Girl Who Stood Up for Education and Was Shot by the Taliban>, <Prod
uct: Batman: The Dark Knight Returns>, <Product: Pride and Prejudice>, <P
roduct: Batman: The Killing Joke>, <Product: Elon Musk: Tesla, SpaceX, an
d the Quest for a Fantastic Future>]
[09/Aug/2019 15:16:23] "GET /recommend/ HTTP/1.1" 200 17695
```

Figure 8. CNN-LSTM model output.

Table 4 and Figure 9 illustrate the overall effectiveness of various sentiment analysis algorithms applied to multiple datasets. In a comparison of sentiment analysis techniques, CNN-LSTM surpassed all of them, achieving accuracy rates of 95, 94, 96, 95, 94, 96, 95, 94, 96, 95, 94, 96, 95, 94, 93, and 96 percent for each dataset.

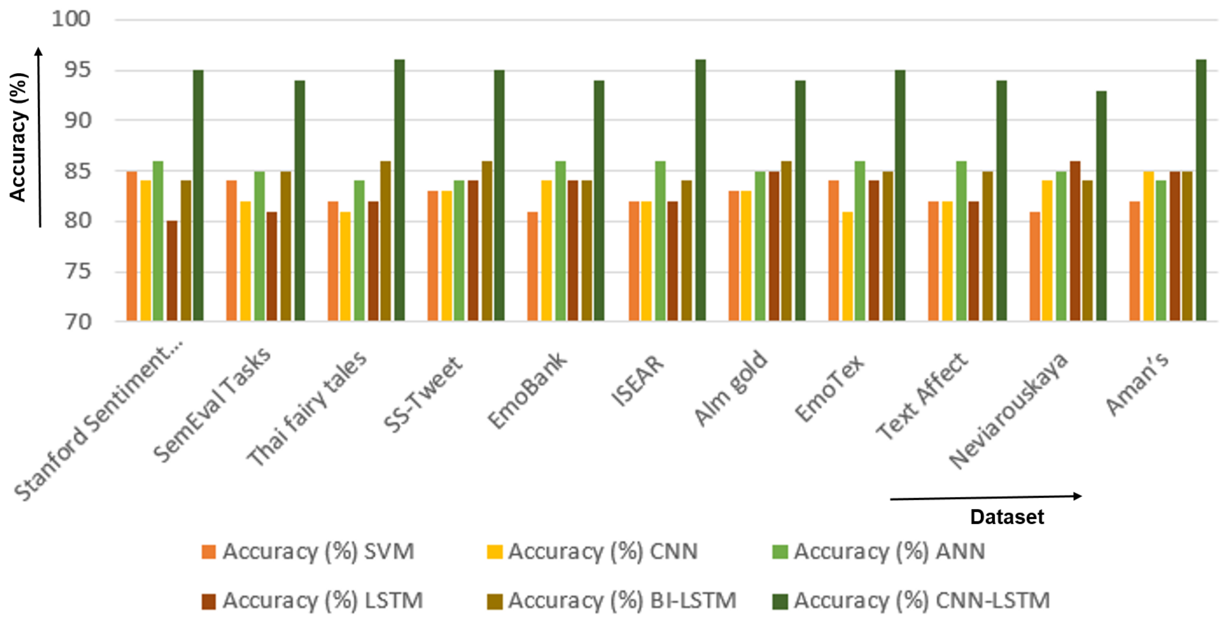


Figure 9. Performance analysis for BOW + IF-IDF with CNN-LSTM based on accuracy. ("Stanford Sentiment..." means "Stanford Sentiment Treebank").

Table 4. Performance analysis for BOW + IF-IDF with CNN-LSTM based on accuracy.

Database Name	Accuracy (%)					
	SVM	CNN	ANN	LSTM	BI-LSTM	CNN-LSTM
Stanford-t Treebank	85	84	86	80	84	95
SemEval Tasks	84	82	85	81	85	94
Thai fairy tales	82	81	84	82	86	96
SS-Tweet	83	83	84	84	86	95
EmoBank	81	84	86	84	84	94
ISEAR	82	82	86	82	84	96
Alm gold	83	83	85	85	86	94
EmoTex	84	81	86	84	85	95
Text Affect	82	82	86	82	85	94
Neviarouskaya	81	84	85	86	84	93
Aman's	82	85	84	85	85	96

Precision: It determines how accurately the model can classify a sample as positive [34]. It is calculated by dividing the total number of positive cases by the number of positively identified positive samples as correct or incorrect.

$$Precision = TP/(TP + FP) \tag{26}$$

Table 5 and Figure 10 illustrate the overall effectiveness of various sentiment analysis algorithms applied to multiple datasets. Comparing sentiment analysis techniques, CNN-LSTM performs better than all, with precision values of 94, 95, 96, 94, 95, 93, 94, 95, 96, and 95 percent for each dataset.

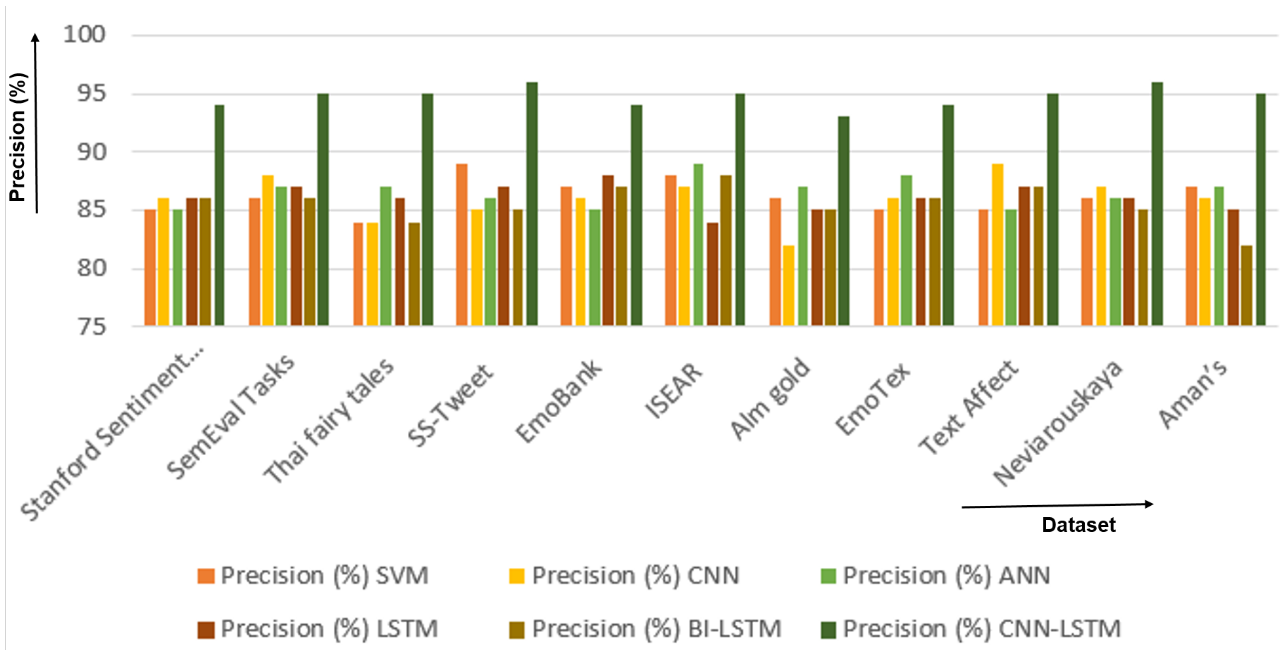


Figure 10. Performance analysis for BOW + IF-IDF with CNN-LSTM based on precision. ("Stanford Sentiment..." means "Stanford Sentiment Treebank").

Recall: This score reflects how well the model can recognize positive samples. The percentage is calculated by the ratio of the total volume of positive samples to the total number of positive samples that were correctly recognized as positive.

$$Recall = \frac{TP}{(TP + FN)} \tag{27}$$

Table 6 and Figure 11 display the overall recall performance of several sentiment analysis algorithms applied to various datasets. CNN-LSTM fared better than all other sentiment analysis techniques when compared, with recall rates of 95%, 94%, 96.5%, 92.5%, 94.5%, 95.5%, 96.5%, 94.5%, 91.5%, 94.5%, and 95% for each dataset.

Table 5. Performance analysis for BOW + IF-IDF with CNN-LSTM based on precision.

Database Name	Precision (%)					
	SVM	CNN	ANN	LSTM	BI-LSTM	CNN-LSTM
Stanford—Treebank	85	86	85	86	86	94
SemEval Tasks	86	88	87	87	86	95
Thai fairy tales	84	84	87	86	84	95
SS-Tweet	89	85	86	87	85	96
EmoBank	87	86	85	88	87	94
ISEAR	88	87	89	84	88	95
Alm gold	86	82	87	85	85	93
EmoTex	85	86	88	86	86	94
Text Affect	85	89	85	87	87	95
Neviarouskaya	86	87	86	86	85	96
Aman's	87	86	87	85	82	95

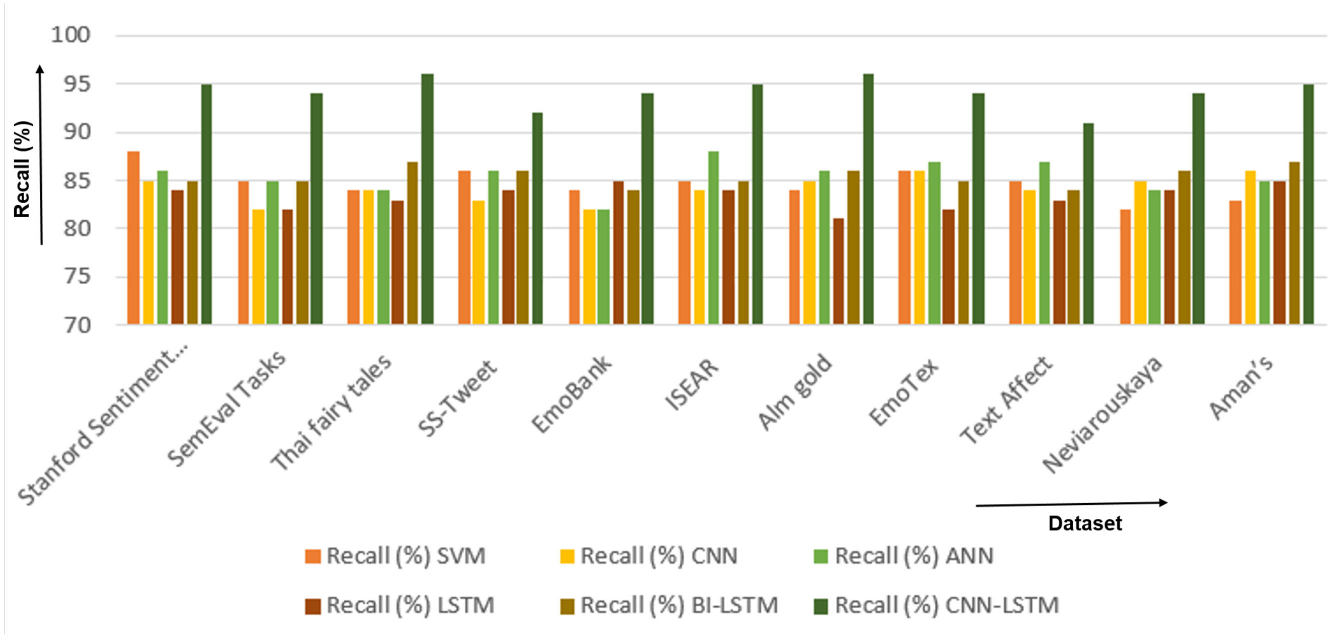


Figure 11. Performance analysis for BOW + IF-IDF with CNN-LSTM based on recall. ("Stanford Sentiment..." means "Stanford Sentiment Treebank").

Table 6. Performance analysis for BOW + IF-IDF with CNN-LSTM based on recall.

Database Name	Recall (%)					
	SVM	CNN	ANN	LSTM	BI-LSTM	CNN-LSTM
Stanford—Treebank	88	85	86	84	85	95
SemEval Tasks	85	82	85	82	85	94
Thai fairy tales	84	84	84	83	87	96
SS-Tweet	86	83	86	84	86	92
EmoBank	84	82	82	85	84	94
ISEAR	85	84	88	84	85	95
Alm gold	84	85	86	81	86	96
EmoTex	86	86	87	82	85	94
Text Affect	85	84	87	83	84	91
Neviarouskaya	82	85	84	84	86	94
Aman's	83	86	85	85	87	95

F1-measure: The F1-measures are estimated from the harmonic mean of accuracy and recall.

$$F1 - measure = \frac{(2 \times precision \times recall)}{(Precision + recall)} \quad (28)$$

$$F1 - Measure = \frac{(2 \times TP)}{((2 \times TP) + FP + FN)} \quad (29)$$

Table 7 and Figure 12 display the overall F1-measure performance of several sentiment analysis methods applied to various datasets. In comparing sentiment analysis techniques, CNN-LSTM fared better than all others, with an F1-measure of 94%, 95%, 94%, 93%, 964%, 95%, 93%, 94%, 92%, 91%, and 94% for each dataset.

Sensitivity: It measures how well the positive class was expected by referring to the proportion of accurately identified real positives [34].

$$Sensitivity = \frac{TP}{(TP + FN)} \quad (30)$$

Table 8 and Figure 13 display the total sensitivity of various sentiment analysis approaches applied to multiple datasets. In comparing sentiment analysis techniques, CNN-LSTM fared better than all other approaches, with sensitivity values for each dataset of 94%, 92%, 93%, 91%, 92%, 93%, 91%, 93%, 94%, 94%, and 93%.

Table 7. Performance analysis for BOW + IF-IDF with CNN-LSTM based on F1-measure.

Database Name	F1-Measure					
	SVM	CNN	ANN	LSTM	BI-LSTM	CNN-LSTM
Stanford—Treebank	86	86	87	85	87	94
SemEval Tasks	85	85	82	84	86	95
Thai fairy tales	84	84	84	84	85	94
SS-Tweet	86	86	86	85	86	93
EmoBank	87	85	89	86	88	96
ISEAR	89	87	87	83	87	95
Alm gold	85	86	85	86	85	93
EmoTex	84	85	86	85	84	94
Text Affect	87	84	84	84	89	92
Neviarouskaya	88	87	86	85	84	91
Aman's	86	86	85	84	85	94

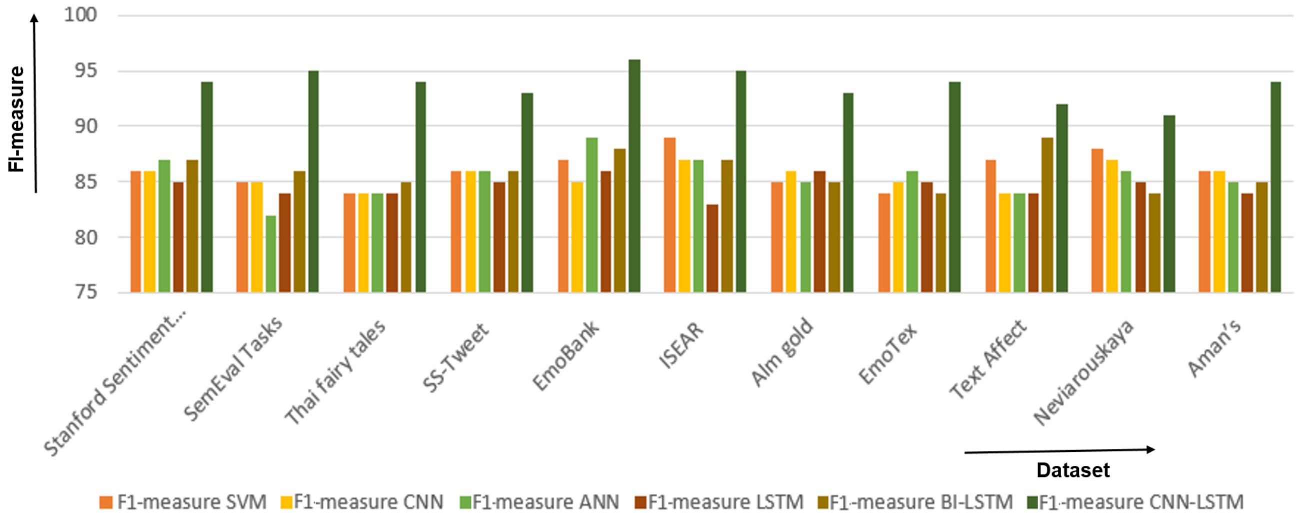


Figure 12. Performance analysis for BOW + IF-IDF with CNN-LSTM based on F1-measure. ("Stanford Sentiment..." means "Stanford Sentiment Treebank").

Specificity: The real negative rate, which is the inverse of sensitivity, is used to assess how accurately the negative class predicted the outcome. An unequal categorization's sensitivity could be more fascinating than its specificity.

$$Specificity = \frac{TN}{(FP + TN)} \quad (31)$$

Table 9 illustrates the overall effectiveness of various sentiment analysis algorithms applied to multiple datasets. With a specificity of 96, 95, 95, 95, 96, 94, 94, 95, 96, and 95 percent for each dataset, CNN-LSTM surpassed all other sentiment analysis techniques in the comparison.

Table 8. Performance analysis for BOW + IF-IDF with CNN-LSTM based on sensitivity.

Database Name	**Sensitivity**					
	SVM	**CNN**	**ANN**	**LSTM**	**BI-LSTM**	**CNN-LSTM**
Stanford—Treebank	85	87	85	84	86	94
SemEval Tasks	86	88	85	85	85	92
Thai fairy tales	87	86	89	82	85	93
SS-Tweet	87	85	87	83	86	91
EmoBank	89	84	86	84	87	92
ISEAR	87	87	84	85	84	93
Alm gold	87	89	89	82	85	91
EmoTex	85	87	87	81	86	93
Text Affect	86	89	88	86	87	94
Neviarouskaya	84	86	89	84	85	92
Aman's	85	87	86	85	84	93

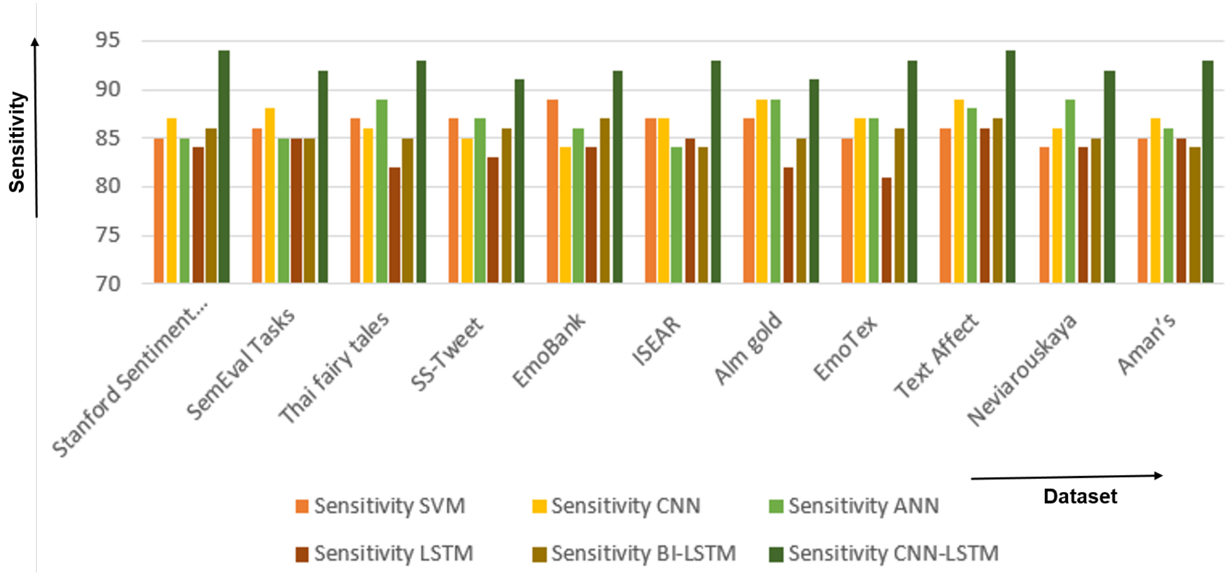


Figure 13. Performance analysis for BOW + IF-IDF with CNN-LSTM based on sensitivity. ("Stanford Sentiment..." means "Stanford Sentiment Treebank").

Table 9. Performance analysis for BOW + IF-IDF with CNN-LSTM based on specificity.

Database Name	**Specificity**					
	SVM	**CNN**	**ANN**	**LSTM**	**BI-LSTM**	**CNN-LSTM**
Stanford—Treebank	82	86	85	85	87	96
SemEval Tasks	85	87	84	84	85	95
Thai fairy tales	84	85	84	86	86	96
SS-Tweet	86	84	85	82	87	95
EmoBank	85	86	84	84	88	96
ISEAR	82	86	86	85	85	94
Alm gold	83	85	83	86	84	96
EmoTex	84	84	84	84	86	94
Text Affect	85	85	85	82	87	95
Neviarouskaya	84	82	86	83	85	96
Aman's	86	87	85	86	86	95

Geometric-mean (G-mean) A single number that strikes a balance between sensitivity and specificity makes up this metric [35].

$$G - mean = \sqrt{(specificity \times sensitivity)} \tag{32}$$

Table 10 displays the overall G-mean performance of various sentiment analysis methods used for multiple datasets. In terms of G-mean, which was 0.916, 0.952, 0.962, 0.957, 0.956, 0.959, 0.958, 0.954, 0.956, 0.957, and 0.949 for each dataset, CNN-LSTM performed better than any other sentiment analysis techniques.

Table 10. Performance analysis for BOW + IF-IDF with CNN-LSTM based on G-Mean.

Database Name	**G-Mean**					
	SVM	**CNN**	**ANN**	**LSTM**	**BI-LSTM**	**CNN-LSTM**
Stanford—Treebank	0.832	0.847	0.882	0.817	0.828	0.916
SemEval Tasks	0.835	0.845	0.885	0.815	0.827	0.952
Thai fairy tales	0.836	0.841	0.886	0.812	0.829	0.962
SS-Tweet	0.835	0.842	0.882	0.812	0.844	0.957
EmoBank	0.832	0.847	0.887	0.822	0.835	0.956
ISEAR	0.832	0.848	0.888	0.827	0.837	0.959
Alm gold	0.834	0.847	0.889	0.828	0.838	0.958
EmoTex	0.832	0.846	0.882	0.819	0.839	0.954
Text Affect	0.837	0.844	0.885	0.817	0.848	0.956
Neviarouskaya	0.832	0.841	0.888	0.841	0.829	0.957
Aman's	0.831	0.843	0.886	0.852	0.836	0.949

5. Conclusions and Future Work

Two current research fields that are closely connected are sentiment analysis and the detection and recognition of emotions from text. Sentiment analysis looks for positive, neutral, or negative textual emotions. Emotion analysis is used to determine whether a text has good, neutral, or negative emotions. This study proposes a four-level technique for recommending the best book to users. Semantic network grouping of similar sentences, sentiment analysis (SA), reviewer clustering, and recommendation system are among the levels. The semantic network uses the parts of speech (POS) tagger to group comparable sentences at the first level, utilizing pre-processed data from reviewer and book databases. BOW and TF-IDF algorithms are used in feature extraction to extract keywords from pre-processed data. At the second level, SA is carried out in two phases: training and testing, using deep learning techniques such as convolutional neural networks (CNN)-LSTM. The findings of this level are transferred to the third level (clustering), which groups the reviewers by age, geography, and gender using the clustering approach. The model is evaluated at the final level using accuracy, precision, recall, sensitivity, specificity, G-mean, and F1-measure. The book recommendation system is made to deliver maximum accuracy in the within minimum number of epochs compared to other methods such as SVM, CNN, ANN, LSTM, and BI-LSTM.

Author Contributions: Conceptualization, S.D.G., and S.K.G.; methodology, A.J. and M.R.; software, S.K.J.; validation, M.R., S.K.J. and S.D.G.; formal analysis, S.K.G.; investigation, A.J.; resources, S.K.J.; data curation, M.R.; writing—original draft preparation, S.D.G.; writing—review and editing, M.R.; visualization, S.K.J.; supervision, M.R.; project administration, A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhai, G.; Yang, Y.; Wang, H.; Du, S. Multi-attention fusion modeling for sentiment analysis of educational big data. *Big Data Min. Anal.* **2020**, *3*, 311–319. [CrossRef]
2. Wong, T.T.; Yeh, P.Y. Reliable accuracy estimates from k-fold cross validation. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1586–1594. [CrossRef]

3. Xu, G.; Yu, Z.; Yao, H.; Li, F.; Meng, Y.; Wu, X. Chinese text sentiment analysis based on extended sentiment dictionary. *IEEE Access* **2019**, *7*, 43749–43762. [CrossRef]
4. Li, Z.; Li, R.; Jin, G. Sentiment analysis of danmaku videos based on naïve bayes and sentiment dictionary. *IEEE Access* **2020**, *8*, 75073–75084. [CrossRef]
5. Wu, J.; Lu, K.; Su, S.; Wang, S. Chinese micro-blog sentiment analysis based on multiple sentiment dictionaries and semantic rule sets. *IEEE Access* **2019**, *7*, 183924–183939. [CrossRef]
6. Wang, Y.; Huang, G.; Li, J.; Li, H.; Zhou, Y.; Jiang, H. Refined global word embeddings based on sentiment concept for sentiment analysis. *IEEE Access* **2021**, *9*, 37075–37085. [CrossRef]
7. Obiedat, R.; Al-Darras, D.; Alzaghoul, E.; Harfoushi, O. Arabic Aspect-Based Sentiment Analysis: A Systematic Literature Review. *IEEE Access* **2021**, *9*, 152628–152645. [CrossRef]
8. Roccetti, M.; Delnevo, G.; Casini, L.; Cappiello, G. Is bigger always better? A controversial journey to the center of machine learning design, with uses and misuses of big data for predicting water meter failures. *J. Big Data* **2019**, *6*, 1–23. [CrossRef]
9. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [CrossRef]
10. Alattar, F.; Shaalan, K. Using artificial intelligence to understand what causes sentiment changes on social media. *IEEE Access* **2021**, *9*, 61756–61767. [CrossRef]
11. Silva, H.; Andrade, E.; Araújo, D.; Dantas, J. Sentiment Analysis of tweets Related to SUS before and during COVID-19 pandemic. *IEEE Lat. Am. Trans.* **2021**, *20*, 6–13. [CrossRef]
12. Januário, B.A.; Carosia, A.E.D.O.; da Silva, A.E.A.; Coelho, G.P. Sentiment analysis applied to news from the Brazilian stock market. *IEEE Lat. Am. Trans.* **2021**, *20*, 512–518. [CrossRef]
13. Phan, H.T.; Tran, V.C.; Nguyen, N.T.; Hwang, D. Improving the performance of sentiment analysis of tweets containing fuzzy sentiment using the feature ensemble model. *IEEE Access* **2020**, *8*, 14630–14641. [CrossRef]
14. Sehar, U.; Kanwal, S.; Dashtipur, K.; Mir, U.; Abbasi, U.; Khan, F. Urdu Sentiment Analysis via Multimodal Data Mining Based on Deep Learning Algorithms. *IEEE Access* **2021**, *9*, 153072–153082. [CrossRef]
15. Smetanin, S. The applications of sentiment analysis for Russian language texts: Current challenges and future perspectives. *IEEE Access* **2020**, *8*, 110693–110719. [CrossRef]
16. Zhang, B.; Xu, D.; Zhang, H.; Li, M. STCS lexicon: Spectral-clustering-based topic-specific Chinese sentiment lexicon construction for social networks. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 1180–1189. [CrossRef]
17. Hou, Q.; Han, M.; Cai, Z. Survey on data analysis in social media: A practical application aspect. *Big Data Min. Anal.* **2020**, *3*, 259–279. [CrossRef]
18. Saad, S.E.; Yang, J. Twitter sentiment analysis based on ordinal regression. *IEEE Access* **2019**, *7*, 163677–163685. [CrossRef]
19. Bie, Y.; Yang, Y. A multitask multiview neural network for end-to-end aspect-based sentiment analysis. *Big Data Min. Anal.* **2021**, *4*, 195–207. [CrossRef]
20. Hu, T.; She, B.; Duan, L.; Yue, H.; Clunis, J. A systematic spatial and temporal sentiment analysis on geo-tweets. *IEEE Access* **2019**, *8*, 8658–8667. [CrossRef]
21. Yu, J.; Jiang, J.; Xia, R. Entity-sensitive attention and fusion network for entity-level multimodal sentiment classification. *ACM Trans. Audio Speech Lang. Process.* **2019**, *28*, 429–439. [CrossRef]
22. Ren, R.; Wu, D. An innovative sentiment analysis to measure herd behavior. *IEEE Trans. Syst. Man. Cybern. Syst.* **2018**, *50*, 3841–3851. [CrossRef]
23. Iqbal, F.; Hashmi, J.M.; Fung, B.C.; Batool, R.; Khattak, A.M.; Aleem, S.; Hung, P.C. A hybrid framework for sentiment analysis using genetic algorithm based feature reduction. *IEEE Access* **2019**, *7*, 14637–14652. [CrossRef]
24. Sánchez-Núñez, P.; Cobo, M.J.; De Las Heras-Pedrosa, C.; Peláez, J.I.; Herrera-Viedma, E. Opinion mining, sentiment analysis and emotion understanding in advertising: A bibliometric analysis. *IEEE Access* **2020**, *8*, 134563–134576. [CrossRef]
25. Mehanna, Y.S.; Mahmuddin, M.B. A Semantic Conceptualization Using Tagged Bag-of-Concepts for Sentiment Analysis. *IEEE Access* **2021**, *9*, 118736–118756. [CrossRef]
26. Zhang, B.; Li, X.; Xu, X.; Leung, K.C.; Chen, Z.; Ye, Y. Knowledge guided capsule attention network for aspect-based sentiment analysis. *ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2538–2551. [CrossRef]
27. Seo, S.; Na, S.; Kim, J. HMTL: Heterogeneous modality transfer learning for audio-visual sentiment analysis. *IEEE Access* **2020**, *8*, 140426–140437. [CrossRef]
28. Kim, R.Y. Using Online Reviews for Customer Sentiment Analysis. *IEEE Eng. Manag. Rev.* **2021**, *49*, 162–168. [CrossRef]
29. Zhu, L.; Li, W.; Shi, Y.; Guo, K. SentiVec: Learning sentiment-context vector via kernel optimization function for sentiment analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 2561–2572. [CrossRef]
30. Zhang, K.; Zhu, Y.; Zhang, W.; Zhang, W.; Zhu, Y. Transfer correlation between textual content to images for sentiment analysis. *IEEE Access* **2020**, *8*, 35276–35289. [CrossRef]
31. Sweidan, A.H.; El-Bendary, N.; Al-Feel, H. Sentence-level aspect-based sentiment analysis for classifying adverse drug reactions (ADRs) using hybrid ontology-XLNet transfer learning. *IEEE Access* **2021**, *9*, 90828–90846. [CrossRef]
32. She, D.; Yang, J.; Cheng, M.M.; Lai, Y.K.; Rosin, P.L.; Wang, L. Wscnet: Weakly supervised coupled networks for visual sentiment classification and detection. *IEEE Trans. Multimed.* **2019**, *22*, 1358–1371. [CrossRef]
33. Basile, V.; Novielli, N.; Croce, D.; Barbieri, F.; Nissim, M.; Patti, V. Sentiment polarity classification at evalita: Lessons learned and open challenges. *IEEE Trans. Affect. Comput.* **2018**, *12*, 466–478. [CrossRef]

34. Huang, F.; Yuan, C.; Bi, Y.; Lu, J. Exploiting long-term dependency for topic sentiment analysis. *IEEE Access* **2020**, *8*, 221963–221974. [CrossRef]
35. Cobos, R.; Jurado, F.; Blázquez-Herranz, A. A content analysis system that supports sentiment analysis for subjectivity and polarity detection in online courses. *IEEE Rev. Iberoam. Tecnol. Aprendiz.* **2019**, *14*, 177–187. [CrossRef]