Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-19086-y

A novel self-supervised sentiment classification approach using semantic labeling based on contextual embeddings

Mousa Alizadeh[1] · **Azam Seilsepour**[2]

Received: 18 October 2023 / Revised: 11 February 2024 / Accepted: 22 March 2024

Abstract
Sentiment Analysis (SA) is a domain or context-oriented task since the sentiment words convey different sentiments in various domains. As a result, the domain-independent lexicons cannot correctly recognize the sentiment of domain-dependent words. To address this problem, this paper proposes a novel self-supervised SA method based on semantic similarity, contextual embedding, and Deep Learning Techniques. It introduces a new Pseudo-label generator that estimates the pseudo-labels of samples using semantic similarity between the samples and their sentiment words. It proposes two new concepts to calculate semantic similarity: The Soft-Cosine Similarity of a sample with its Positive words (SCSP) and the Soft-Cosine Similarity of a document with its Negative words (SCSN). Then, the Pseudo-label generator uses these concepts and the number of sentiment words to estimate the label of each sample. Later on, a novel method is proposed to find the samples with highly accurate pseudo-labels. Finally, a hybrid classifier, composed of a Convolutional Neural Network (CNN) and a Gated Recurrent Unit (GRU), is trained using these highly accurate pseudo-labeled data to predict the label of unseen data. The comparison of the proposed method with the lexicons and other similar existing methods demonstrates that the proposed method outperforms them in terms of accuracy, precision, recall, and F1 score.

Keywords Topic Sentiment Analysis · Self-supervised Learning · Semantic Similarity · Contextual Embedding

1 Introduction

Due to the rapidly growing access to the Internet, a huge amount of data, such as text, images, and videos, is generated freely every day on social media platforms such as Facebook, Twitter,

✉ Mousa Alizadeh
s3970482@student.rmit.edu.au

Azam Seilsepour
Aza.seilsepour.eng@iauctb.ac.ir

[1] School of Engineering, RMIT University, Melbourne, VIC, Australia

[2] Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran

Published online: 13 May 2024

and Instagram. This data expresses social media users' feelings, opinions, and experiences about events, topics, services, and products. Analyzing this data helps politicians make better decisions, and the consumers and service providers develop business strategies and improve their products and services [1]. As a result, Natural Language Processing (NLP) is getting more attention every day, and Sentiment Analysis (SA), a branch of NLP that concentrates on finding the sentiment orientation of a sentence or a document, is getting increasingly popular [2, 3].

SA approaches commonly fall into Machine Learning (ML) and knowledge-based approaches. Knowledge-based methods usually employ general-purpose lexicons or knowledge graphs to find the sentiment orientation of sentences [4, 5]. Both kinds of approaches have their pros and cons. Knowledge-based approaches do not need labeled data. They are computationally effective and scalable. However, they cannot detect the labels correctly when the margin between the labels is too small, the samples are short, or the data is noisy, complex, and ambiguous. Additionally, their performance varies remarkably in different domains [6, 7]. On the other hand, ML-based approaches require labeled data, and providing the labeled data is commonly expensive and time-consuming. As a result, remarkable research attention has been paid to hybrid approaches that combine knowledge- and ML-based methods using a two-staged pipeline. First, a general-purpose lexicon is utilized to classify the samples. Second, a classifier is trained by these labeled samples to predict the sentiment of unseen data [8–15].

However, the hybrid approaches suffer from not correctly recognizing the sentiments of domain- or context-dependent words since SA is a domain- or context-dependent task. The sentiment of words varies in different domains. For instance, the "low price" conveys positive sentiment in a product review, but the "low salary" conveys negative sentiment in a job description. In another instance, the adjective "easy" has a positive sentiment in the phrase "easy to use" in a software product review but has a negative sentiment in the phrase 'easy game' in a computer game review. As a result, a domain-independent lexicon can not recognize the sentiment of domain-dependent words. Additionally, an ML model trained on a specific domain can not be utilized in another domain [16, 17].

In recent years, self-supervised methods, a kind of unsupervised method, have been employed to solve these problems. These methods automatically generate pseudo-labels using the contents of samples. They borrow a list of sentiment words and their sentiments from the lexicons, extract the sentiment words of the samples, and estimate the pseudo-labels. Later on, a classifier is trained using these pseudo-labeled samples to predict the sentiments of unseen data. These methods widely use the number of positive and negative words corresponding to each sample to estimate the pseudo-labels. For instance, Sazzed et al. [9] and Rendon et al. [18] used the number of positive and negative words of each sample to estimate the appropriate pseudo-labels and trained the SVM and LR classifiers with these pseudo-labeled data. However, these methods utilize domain-independent lexicons, so they can not recognize the sentiment of domain-dependent words.

To address the previously described problem not correctly recognizing the sentiments of domain- or context-dependent words, this paper proposes a novel self-supervised SA approach that does not need any labeled data and considers the context of samples to generate pseudo-labels. To do this, the proposed method offers a semantic-based pseudo-label generator that estimates the pseudo-label of samples using contextual embeddings and semantic similarity between the context of samples and their corresponding sentiment words. It uses two newly introduced concepts: Soft-Cosine Similarity [19] of a sample with its Positive words (SCSP) and Soft-Cosine Similarity of a document with its Negative words (SCSN). The Soft-Cosine similarity is a text similarity measure that calculates the semantic simi-

larity between two sentences, even if they have no common words but the same meaning. The semantic-based pseudo-label generator converts all the words into dense feature vectors, calculates SCSP and SCSN, and estimates the pseudo-labels. Additionally, when the SCSP and SCSN are equal, another two concepts are calculated and used: Cosine Similarity [20] of a document with its Positive words (CSP) and Cosine Similarity of a document with its Negative words (CSN). Later on, a new method is proposed to find the samples with highly accurate pseudo-labels. Finally, a hybrid classifier, composed of Convolutional Neural Network (CNN) [21], and Gated Recurrent Unit (GRU) [22] is trained with these pseudo-labeled samples. To the best of our knowledge, it is the first time that the semantic similarity, contextual embeddings, and the number of sentiment words are considered jointly to estimate the pseudo-labels based on the context of samples.

The main contribution of this paper can be summarized as follows:

- Proposing a self-supervised SA method that does not require labeled data.
- Proposing a novel semantic-based pseudo-label generator that estimates the pseudo-labels of samples based on semantic similarity and the number of sentiment words.
- Proposing a hybrid sentiment classifier composed of CNN and GRU model.

The rest of the paper is structured as follows: Section 2 explains the research basics, which are the proposed method's building blocks. Section 3, Literature Review, describes the previous similar methods, Section 4 explains the proposed method in detail, and Section 5 contains metrics and evaluation results. Finally, Section 6 consists of the conclusion and future work.

2 Basics of research

In this section, the building blocks of the proposed method, including Document Embedding, Soft Cosine similarity, CNN, and GRU are explained in detail.

2.1 Document embedding

ML algorithms need their input to be represented as fixed-length feature vectors. Bag-of-words (BOW) and bag-of-n-grams are widely used methods to convert texts into fixed-length feature vectors, but they do not capture the word order. As a result, the sentences composed of the same words in different orders have the same representations. Bag-of-n-grams capture the order of words, but they have the problem of data sparsity and high dimensionality. Additionally, these algorithms do not capture the distance between words correctly. In other words, they do not consider the semantics of words. For instance, the words "powerful", "strong", and "Paris" have the same distance, while "powerful" is semantically closer to "strong" [23]. Paragraph Vector or Doc2Vec, inspired from [24], is a framework that converts every sentence, paragraph, or text of different lengths into fixed-length feature vectors. This method concatenates or averages the word vectors to predict the next word in the sentence. It works based on two different modes: the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words of Paragraph Vectors (PV-DBOW). Like the continuous bag of words, the former is more complex but performs better. The PV-DM method either concatenates or averages all word embeddings of a document to calculate document embeddings. Like skip-gram, the latter is simpler and usually leads to a higher error rate [23].

2.2 Cosine and soft-cosine similarity

Calculating the similarity of texts is essential in various tasks of NLP, such as question answering, plagiarism detection, SA, etc. The Cosine similarity [20] is widely used to measure the similarity between the texts. It calculates the Cosine of the angle between the feature vectors. To use the Cosine similarity, each text should be represented as a vector of feature values, and each feature corresponds to a dimension in the Vector Space Model (VSM). In the field of NLP, the most widely used features are words and n-grams. The Cosine similarity is calculated as below:

$$Cosine(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum A_i^2} \times \sqrt{\sum B_i^2}} \tag{1}$$

Where A and B are two vectors. However, the Cosine similarity does not consider the number of features the texts share and the number of zero features. In other words, it does not measure the similarity between the features of vectors, which are the words, in the context of NLP [25]. For instance, consider the following sentences:
a: a player will play a game they like to play
b: they play the game they like
The bag-of-words representations of a and b are the following vectors:
a = (2, 1, 1, 2, 1, 1, 1, 1, 0)
b = (0, 0, 0, 1, 1, 2, 1, 0, 1)
where the values indicate the number of words a player will play in a game, they like, too, and the in sentences. The Cosine similarity of a and b, based on (1), is zero [19]. Soft-Cosine similarity [19] is another semantic measure that considers the similarity of features and is calculated as below:

$$Soft - Cosine(A, B) = \frac{\sum \sum_{i,j}^{N} s_{ij} a_i b_j}{\sqrt{\sum \sum_{i,j}^{N} s_{ij} a_i a_j} \times \sum_{i,j}^{N} s_{ij} b_i b_j} \tag{2}$$

$$s_{ij} = cosine(e^i, e^j) \tag{3}$$

Where e^i and e^j are the primary vectors of a_i and b_j. When the a_i and b_j are completely different, the s_{ij} will be 0. As shown in (2), the Soft-Cosine similarity calculates the similarity between the features.

2.3 Convolutional neural network

Deep learning is a sub-area of ML algorithms inspired by artificial neural networks. A Deep Neural Network (DNN) is a sequence of layers that learns data representations. Since DNNs can automatically identify and extract text features, they are increasingly used in various NLP tasks, such as SA. They are inspired by the structure of the human brain and consist of a large number of information processing units, called neurons, are organized in a sequence of layers. They can learn to perform tasks such as regression and classification by adjusting the connection weights between neurons, mimicking the learning process of a human brain [26].

Among the different types of DNNs, Recurrent Neural Networks (RNN) and CNNs have been widely used in SA. It has been proven that CNNs can improve the accuracy of text classification since they extract local and deep features [27]. CNNs are feed-forward neural

networks composed of three layers: convolution, pooling, and fully connected. The convolution layer extracts the features, and the pooling layer reduces the features. The convolution layer applies different filters on embeddings to perform feature selection and create feature maps. The pooling layer reduces the computational workload and speeds up operations for the next layers. The last layer is a fully connected neural network, which consists of an activation function and relates the text or image features to target classes [21].

2.4 Gated recurrent unit

The GRU and Long Short-Term Memory (LSTM) models, a kind of RNNs, have been proposed to solve the vanishing gradient problem. The GRU is similar to the LSTM but does not have a memory cell [28, 29]. It has a simpler architecture and has shown better results in different NLP tasks such as text classification [22]. It uses two gates. The Update gate, indicated by Z_t, decides the amount of data that needs to be kept in the future. On the other hand, the Reset gate, indicated by r_t, decides the amount of data that can be forgotten. The $h_{(t-1)}$ includes the data of the previous state, and the $\hat{h_t}$ determines the data that should be removed from the previous state. The following equations show how the GRU network works. In these equations, σ represents the $sigmoid$ function, and $\odot$ means element-wise multiplication [30–32].

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{4}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{5}$$

$$\hat{h_t} = tanh(W X_t + U(r_t \odot h_{t-1})) \tag{6}$$

$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h_t} \tag{7}$$

3 Literature review

In this section, first, we discuss some unsupervised and self-supervised SA methods. Second, a review of hybrid SA methods is conducted.

3.1 Unsupervised and Self-supervised SA methods

In general, unsupervised methods rely on statistical features of the document, such as word co-occurrence or the presence of sentiment words. In contrast, as shown in Fig. 1, self-supervised methods are a subset of unsupervised learning in which the output labels are generated automatically by extracting patterns from data.

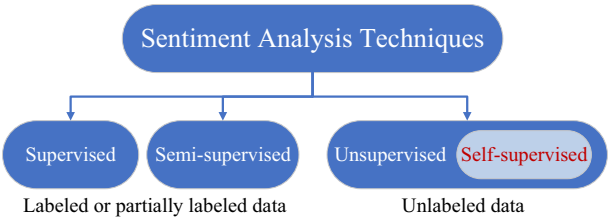

Fig. 1 Sentiment Analysis Methods

A Self-supervised and syntax-based method (SESS) was proposed in [33] that firstly calculated the sentiment score of each document using the positive and negative seeds provided by Subjclueslen1-HLTEMNLP05 [1] dictionary, and the list of seeds was updated iteratively in each step. Secondly, a Naïve Bayes classifier was trained using these labeled documents. In the last step, the train Naïve Bayes (NB) classifier was applied to all datasets to find the labels of documents. Additionally, to improve the quality of labels, they found three types of compound and complex sentences, i.e., coordination, concession, or condition, and considered their sentiment while calculating the sentiment score of documents. SESS was evaluated on Amazon product review dataset [34].

Qiu et al. [35] proposed a Self-Supervised Model for Sentiment Classification (SELC) of the Chinese dataset that includes two steps. First, the HowNet [2] dictionary and a negation list were employed to classify the reviews. Second, a Support Vector Machine (SVM) classifier was trained by these labeled samples. Consequently, this classifier is used to predict the sentiment of unseen reviews. The authors have used the TF-IDF method to make feature vectors of texts which does not consider the semantic meaning of words. Additionally, they have evaluated SELC on the reviews concern with ten domains [3]: Monitors, Mobile phones, Digital Cameras, MP3 players, Computer parts, Video cameras and lenses, Networking, Office equipment, Printers, Computer peripherals. He and Zhou [36] proposed a self-supervised method that borrowed a list of sentiment words from the MPQA lexicon and trained a classifier on the Amazon Review [34] and the Cornel Movie Review datasets [37].

Zhou et al. [38] proposed an unsupervised method called graph co-regularized non-negative matrix tri-factorization (GNMTF) from the geometric perspective. GNMTF assumes that if two words (or documents) are sufficiently close to each other, they have the same sentiment. They constructed the nearest neighbor graphs in conjunction with a non-negative matrix tri-factorization framework.

In [39], the authors propose a lexicon-based method called SmartSA to predict sentiments. This method extracts sentiment from sentiment lexicons. It was proved by analysis and experimental observations that this method works well and with better performance than the SentiStrength [40] method. Jimenz et al. [41] presented an unsupervised Aspect-based sentiment classification method. First, they extract different aspects of each entity. Second, they used Bing Liu, MPQA, and SentiWordNet [42] lexicons to extract the sentiment corresponding to each aspect.

Fernandez et al. [43] proposed an unsupervised dependency parsing-based text classification method that borrows a list of seeds from SO-CAL. Then, it uses linguistic rules to find the sentiments. They evaluated their method on Cornell Movie Review [37], Obama-McCain Debate, and SemEval-2015 dataset [4].

Vilares et al. [44] proposed an unsupervised SA method based on the compositional syntax-based rule. They borrowed a list of seeds as prior knowledge from SO-CAL and evaluated their method on Cornell Movie Review [37], German, and Spanish datasets.

Vanishta and Suzan [45] proposed an unsupervised method based on fuzzy logic that includes four major steps: tokenization, formulation of a bag of words model, formulation of fuzzy sentiment score, and assigning polarity. They have calculated the cardinality of positive and negative words using SentiWordNet [42] and AFINN [46] dictionaries separately. When

[1] https://www.cs.pitt.edu/mpqa/

[2] http://www.keenage.com/download/sentiment.rar

[3] http://www.informatics.sussex.ac.uk/users/tz21/coling08.zip

[4] http://alt.qcri.org/semeval2015/task10/

the cardinality of positive words is equal to or greater than negative words, the label of the document is considered positive. When the cardinality of positive words is less than negative words, the label of the document is considered negative. They have evaluated the proposed method on polarity dataset v2.0 by Pang and Lee1 [37] and IMDB [47]. The third dataset provides reviews of a single hotel [5].

Also, Sazzed et al. [9] proposed a method called SSentiA and used an opinion lexicon to generate pseudo-labels. Then, they utilized these labeled data to train the SVM and LR classifiers and predicted the sentiment of the unseen data. Rendon-Cardona et al. [18] have extended the SSentiA self-supervised method by adding a translation module for sentiment analysis of Spanish texts so that the module can translate Spanish texts into English texts with higher accuracy and performance.

Seilepour et al. [5] used the sum of Cosine and Soft Cosine similarities between the samples and their corresponding sentiment words to estimate the pseudo-labels. Later on, they trained a RoBERTa-GRU classifier using these labeled data. Additionally, they have used the Whale Optimization Algorithm to fine-tune the hyperparameters of the GRU network. In another work, Seilsepour et al. [15] employed the semantic similarity and WMD distance [48] simultaneously to estimate the pseudo-labels. Additionally, they trained a RoBERTa-LSTM classifier with the samples having highly accurate pseudo-labels.

3.2 Hybrid SA methods

Hybrid SA methods, a subset of unsupervised methods, utilize a lexicon to find the appropriate labels corresponding to each sample. Later on, these labeled samples train an ML method to predict the label of unseen data. For instance, An entity-level sentiment analysis method was proposed in [49], which trained an SVM classifier to predict the sentiment of unseen data and a vocabulary-based approach for document labeling.

Iqbal et al. [8] employed SentiWordNet [42] to find the appropriate labels of samples and Bag of Words (BOW) to make feature vectors. Later on, a Genetic Algorithm (GA) was used to reduce the number of features. Finally, these sample data trained the Naive Bayes (NB) classifier. The proposed method was performed on IMDB [47], Yelp [6], and Amazon review datasets [34].

Aljedaani et al. [10] utilized TextBlob [50] to calculate the sentiment score of reviews about US Airlines [7]. They used the Bag of Word and TF-IDF to make feature vectors. Later on, they trained ML models, such as Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), Extra Trees Classifier (ETC), Support Vector Classifier (SVC), and DNN models, such as CNN, GRU, LSTM, LSTM-GRU, and CNN-LSTM. The LSTM-GRU and LSTM achieved the highest accuracy. Azlinah et al. [12] used VADER [51] to find the appropriate labels and Word2vec and Glove to create feature vectors. In the next step, they trained SVM, CNN, LSTM, CNN-LSTM, and LSTM-CNN classifiers. The comparison results showed that CNN outperformed the other methods. Khan et al. [13] estimated the labels using several lexicons such as AFFIN, GL, OL, SentiWordNet, So-CAL, Subjectivity lexicon, WordNet-Affect, NRC, SenticNet5, and SentiSense. In the next step, they created sentence embeddings using BERT and trained a hybrid network composed of a BILSTM and CNN to classify unseen data.

[5] http://www.kaggle.com/harmanpreet93/hotelreviews

[6] https://www.yelp.com/

[7] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Mardjo et al. [11] collected 3.5 million tweets and used VADER [51] to estimate the labels. Later on, they used TF-IDF to create feature vectors and trained an RF classifier. Additionally, they used Grey Wolf Optimizer (GWO) to fine-tune the hyperparameters of the RF classifier.

Kathuria et al. [14] estimated the labels for the feedback of postgraduate students using the SentiWordNet. Additionally, they used TF-IDF to make feature vectors. Later on, they trained the classifiers such as SVM, Multinominal Naive Bayes (MNB), LR, RF, DT, and K-Nearest Neighbor (KNN). The comparison results showed that the RF performed better.

Table 1 shows the list of SA methods.

4 Proposed method

Since SA is a domain- or context-dependent task, the sentiment of words varies in different domains. For instance, the word "unpredictable" conveys a positive sentiment in the phrase "unpredictable plot" in the movie review context but a negative sentiment in the phrase "unpredictable steering". Hence, the SA approaches based on a domain-independent lexicon or an ML model trained on a specific domain can not recognize the sentiment of domain-dependent words correctly [16]. To address this problem, this paper proposes a novel hybrid self-supervised SA approach that does not need labeled data. The proposed method offers a semantic-based pseudo-label generator that captures the semantic relationships between the samples and their corresponding sentiment words and the number of sentiment words to estimate the pseudo-labels. The proposed method utilizes the Cosine [20] and Soft Cosine [19] similarity measures to capture the semantic similarity. As described in Section 2.2, the Cosine similarity is widely used to calculate the semantic similarity between the texts, but it does not consider the similarity between the features of vectors. On the other hand, the Soft Cosine similarity obtains the similarity between the features, too. As a result, the proposed method uses the Cosine and Soft Cosine. It introduces four new semantic concepts: Soft Cosine Similarity of a document with its Positive words (SCSP), Soft Cosine Similarity of a document with its Negative words (SCSN), Cosine Similarity of a document with its Positive words (CSP), and Cosine Similarity of a document with its Negative words (CSN). The semantic-based pseudo-label generator borrows a list of sentiment words from Opinion Lexicon [55] and calculates the SCSP and SCSN. When the SCSP is bigger than SCSN, and the number of positive words is bigger than the number of negative words, the pseudo-label is considered positive. When the SCSP is less than the SCSN and the number of positive words is less than the number of negative words, the pseudo-label is considered negative. In other cases, the CSP and CSN are calculated using the Doc2vec embedding techniques. When the CSP is bigger than the CSN, the pseudo-label is considered positive, and vice versa. The semantic-based pseudo-label generator also introduces a novel method to find highly accurate pseudo-labels. Later on, the samples with highly accurate pseudo-labels are fed into a CNN-GRU classifier. Fig. 2 shows that the proposed method includes four steps: Preprocessing, Semantic Pseudo-label Generator, Finding highly accurate pseudo-labels, and CNN-GRU classifier. In the following subsections, each step is explained in more detail.

4.1 Preprocessing

The preprocessing step is essential in NLP tasks since social media comments are usually full of links, emoticons, etc, and some SA methods are sensitive to errors and mistakes in user-

Table 1 Unsupervised (U), Self-Supervised (S), and Hybrid (H) SA methods

Authors	Type	Lexicons	Details	Datasets
Qiu et al. [35]	S	HowNet	TF-IDF, SVM	Chinese product reviews
Zhang et al. [33]	S		TF-IDF, NB	Amazon product reviews [34]
He and Zhou. [36]	S	MPQA	TF-IDF, NB, SVM, Maximum Entropy	Movie review [37], Amazon product review [34]
Zhang et al. [49]	S	Opinion Lexicon	SVM	Tweets[a]
Zhang et al. [52]		HowNet, Tsinghua	TF-IDF, Enhanced Self-Training (EST)	IT product review [b], Hotel review [c]
Zhou et al. [38]	U	Bing Lu Lexicon [53]	TF-IDF	Movie review [37], Amazon product reviews [34]
Jimenz et al. [41]	U	Bing Liu Lexicon, MPQA and SentiWord-Net	TF-IDF	SemEval 2014-task 4 [d]
Fernandez et al. [43]		Dependency Parsing-based	SO-CAL	Movie review [37], Obama-McCain Debate [e], SemEval-2015 [f]
Muhammad et al. [39]	S	SWN	TF-IDF	Twitter [g], Digg [h], MySpace [i]
Vilares et al. [44]		SO-CAL	TF-IDF	Movie review [37], Product review [54]
Iqbal et al. [8]	S	SentiWordNet	BOW, NB, GA	IMDB [47], Yelp [j], Amazon product reviews [34]
Vanishta and Suzan [45]		SentiWordNet and AFINN	BOW	Twitter [k]
Aljedaani et al. [10]	S	TexBlob	TF-IDF, RF, DT, LR, ETC, SVC, GRU, LSTM, LSTM-GRU, CNN-LSTM	US Airline reviews [l]
Mardjo et al. [11]	S	VADER	TF-IDF, RF, GWO	Twitter [m]
Sazzed et al. [9]	S	Opinion lexicon	TF-IDF, LR, SVM	Movie review [37], Amazon product review [34], TripAdvisor [n]
Rendon-Cardona et al. [18]	S	Opinion lexicon	TF-IDF,LR, SVM, RF, NB	CorpusCine [o], PaperReviews [p]
Seilsepour et al. [5]	S	Opinion lexicon	Word2vec, Doc2vec, RoBERTa-GRU	Movie review [37], Amazon product review [34]

Table 1 continued

Authors	Type	Lexicons	Details	Datasets
Kathuria et al. [14]	S	SentiWordNet	SVM, MNB, LR, RF, DT, KNN	Student feedback
Azlinah et al. [12]	S	VADER	Word2Vec, Glove, SVM, CNN, LSTM, CNN-LST, LSTM-CNN	Tweets [q]
Khan et al. [13]	S	AFFIN, GL, OL, SentiWordNet, So-CAL, MPQA, WordNet-Affect, NRC, Sentic-Net5, SentiSense	BERT, BILSTM, CNN	Movie reviews [37], RT-2K, Stanford Sentiment Treebank [r], Customer review, SemEval 2013, SemEval 2014 [s]
SSTSA [15]	Self-S	Opinion Lexicon	Word2vec, Doc2vec, RoBERTa-LSTM	Movie reviews [37], Amazon product review [34]

[a] https://twitter.com/
[b] http://www.it168.com/
[c] http://www.searchforum.org.cn/tansongbo/corpus/ChnSentiCorp_htl_ba_4000.rar
[d] https://alt.qcri.org/semeval2014/
[e] https://bitbucket.org/speriosu/updown/downloads
[f] http://alt.qcri.org/semeval2015/task10/
[g] http://twitter.om
[h] https://digg.com/
[i] https://myspace.com/
[j] https://www.yelp.com/
[k] http://www.twitter.com
[l] https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[m] https://twitter.com/
[n] https://figshare.com/articles/dataset/TripAdvisor_reviews_of_hotels_and_restaurants_by_gender/6255284
[o] http://www.lsi.us.es/~fermin/corpusCine.zip
[p] https://archive.ics.uci.edu/ml/datasets/Paper+Reviews
[q] https://twitter.com/
[r] https://nlp.stanford.edu/sentiment/treebank.html
[s] https://alt.qcri.org/semeval2014/

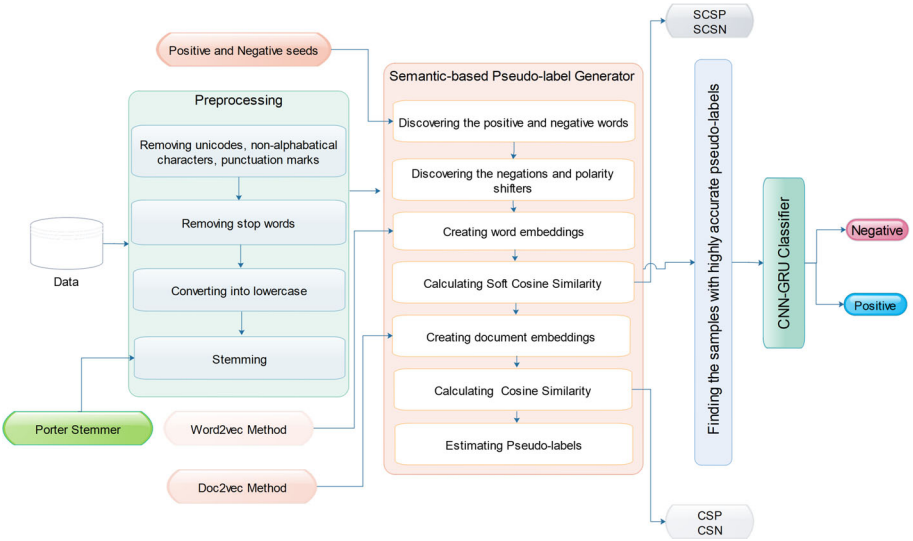

Fig. 2 Architecture of Proposed Method

generated content [15]. This step removes all non-alphabetical characters, links, Unicode, punctuation marks, and stop words. Moreover, all characters will be converted into lowercase, and the tokens will be stemmed by Porter Stemmer [8].

4.2 Semantic-based pseudo-label generator

As explained earlier, the semantic pseudo-label generator captures the semantic relationships between the samples and their corresponding sentiment words. In the following sections, each step is explained in more detail.

4.2.1 Discovering the sentiment words

The semantic-based pseudo-label generator borrows a list of sentiment words and their sentiments from the Opinion Lexicon [55], including 4783 negative and 2006 positive terms. This domain-independent word list is employed as seeds to extract the sentiment words of each sample. For instance, "Nice" conveys positive sentiment in all domains. We suppose that the dataset DS is a collection of N samples, $S_1, S_2, S_3, ..., S_N$. PW_i and NW_i are positive and negative words of sample S_i, respectively. DS, PW_i, and S_i are defined as below:

$$DS = \{S_1, S_2, S_3, ..., S_N\} \tag{8}$$

$$PW_i = \{PW_1, PW_2, PW_3, ..., PW_{nPW}\} \tag{9}$$

$$NW_i = \{NW_1, NW_2, NW_3, ..., NW_{nNW}\} \tag{10}$$

Where nPW_i and nNW_i are the numbers of positive and negative words of S_i, respectively.

[8] https://tartarus.org/martin/PorterStemmer/

4.2.2 Discovering the negations and polarity shifters

The negations like no, never, and not reverse the polarity of their successive word. For instance, "The taste of food is not good" conveys a negative sentiment because of not. In addition, the models, such as should and could reverse sentences' sentiment orientation. For instance, "The quality of food could be better" conveys a negative sentiment [9]. As a result, the successive terms of negations and polarity shifters for each sample S_i are found and appended to the opposite sets of PW_i and NW_i, positive words to NW_i, and vice versa.

4.2.3 Calculating SCSP and SCSN

In this step, the semantic similarity of each sample S_i with its positive and negative words is calculated individually. To this end, the semantic-based pseudo-label generator uses the Soft-Cosine Similarity [19] measure, an extended version of Cosine similarity that considers the similarity between features of sentences. As explained in Section 2.2, Soft-Cosine calculates the similarity between two sentences even if they share no common words. The Soft-Cosine similarity is calculated according to (2). The Soft-Cosine similarity of S_i with its positive and negative sentiment words are defined below:

$$SCSP(S_i) = \sum_{j=1}^{nPWi} Soft - Cosine(S_i, PW_j), \forall i \in \{1, 2, ..., N\} \tag{11}$$

$$SCSN(S_i) = \sum_{j=1}^{nNWi} Soft - Cosine(S_i, NW_j), \forall i \in \{1, 2, ..., N\} \tag{12}$$

Where the $SCSP(S_i)$ is the Soft-Cosine Similarity of S_i with its Positive words, and the $SCSN(S_i)$ is the Soft-Cosine Similarity of S_i with its Negative words. To calculate the Soft-Cosine similarity, all words of samples should be converted into feature vectors. Here, we use the Word2Vec [24] technique. It is a commonly used technique that works based on the theory of information, supposing the words used together convey similar meanings. It uses a shallow neural network to create dense feature vectors of words in a simple and reasonably fast way.

4.2.4 Estimating the pseudo-labels

Here, the pseudo-label of each sample S_i is estimated based on the Soft Cosine similarity between its sentiment words and the sample S_i itself and the number of its positive/negative words. The sentiment orientation of sample S_i is determined by:

$$Pseudo - label(S_i) = \begin{cases} 1 & if\ SCSP(S_i) > SCSN(S_i)\ and\ nPT(S_i) > nNW(S_i) \\ 0 & if\ SCSP(S_i) < SCSN(S_i)\ and\ nPT(S_i) < nNW(S_i) \\ tie & Otherwise \end{cases} \tag{13}$$

Where nPW(S_i) and nNW(S_i) are the numbers of positive and negative words of S_i, as denoted in (9) when the Soft Cosine similarity of S_i with positive words is bigger than the Soft Cosine similarity of S_i with negative words, and the number of positive terms is bigger than the number of negative words, the pseudo-label of S_i is considered as 1 (positive), and vice versa. However, in some cases, the Soft Cosine similarity and the number of sentiment terms do not match. In these cases, the Cosine similarity measure that calculates the Cosine of angles between two vectors is utilized. The cosine between two document vectors, A and B, is calculated according to (1). The cosine similarity of S_i with its positive and negative words is defined as below:

Algorithm 1 Semantic-based Pseudo-label Generator

input : Preprocessed Dataset, Positive seeds, Negative seeds
output: Preprocessed Dataset
Add pseudo_label column to Preprocessed Dataset;
foreach *document* $s \in$ *Preprocessed Dataset* **do**
  $PW =$ *positive words of s using positive seeds*;
  $NW =$ *negative words of s using negative seeds*;
  $negations =$ *negations of s*;
  $polarity-shifters =$ *polarity shifters of s*;
  Add successive words of negations to reverse PW or NW;
  Add successive words of $polarity-shifters$ *to reverse PW or NW*;
  $SCSP = SCSN = 0$;
  $nPW = len(PW)$;
  $nNW = len(NW)$;
  foreach $p \in PW$ **do**
    $SCSP = SCSP + Soft\_Cosine(s, p)$;
  end
  foreach $n \in NW$ **do**
    $SCSN = SCSN + Soft\_Cosine(s, n)$;
  end
  if $SCSP > SCSN$ ***and*** $nPW > nNW$ **then**
    $pseudo\_label \leftarrow 1$;
  else if $SCSP < SCSN$ ***and*** $nPW < nNW$ **then**
    $pseudo\_label \leftarrow 0$;
  else
    $CSP = CSN = 0$;
    foreach $p \in PW$ **do**
      $CSP = CSP + Cosine(p, s)$;
    end
    foreach $n \in NT$ **do**
      $CSN = CSN + Cosine(n, s)$;
    end
    if $CSP >= CSN$ **then**
      $pseudo\_label \leftarrow 1$;
    else
      $pseudo\_label \leftarrow 0$;
    end
  end
  insert *pseudo_label in Preprocessed Dataset*;
end
return *Preprocessed Dataset*;

$$CSP(S_i) = \sum_{j=1}^{nPWi} Cosine(S_i, PW_j), \forall i \in \{1, 2, ..., N\} \tag{14}$$

$$CSN(S_i) = \sum_{j=1}^{nNWi} Cosine(S_i, NW_j), \forall i \in \{1, 2, ..., N\} \tag{15}$$

Where the CSP(S_i) is the Cosine Similarity of S_i with its positive words, and the CSN(S_i) is the Cosine Similarity of S_i with its negative words. The pseudo-label of tie is calculated as below:

$$Pseudo-label(tie) = \begin{cases} 1 & if \quad CSP(S_i) \geq CSN(S_i) \\ 0 & if \quad CSP(S_i) < CSN(S_i) \end{cases} \tag{16}$$

As shown in (16), when the Cosine similarity of the S_i with its positive terms is bigger than its Cosine similarity with its negative terms, the pseudo-label is considered 1 (Positive), and vice versa. All dataset samples should be converted into dense document feature vectors to calculate the Cosine similarity. Here, we use the Doc2Vec [23] technique to convert the

samples of datasets into document feature vectors. As described in Section 2.1, it averages or concatenates the feature vectors of words composing a sample text or paragraph to create the document feature vectors. We use the word feature vectors created in Section 4.2.3 to create the document feature vectors using Doc2Vec in a simple and reasonably fast way.

Algorithm 1 shows the process of generating pseudo-labels in more detail. As shown in Algorithm 1, first, sentiment words, negations, and polarity shifters of documents are extracted separately. Then, SCSP, SCSN, CSP, and CSN are calculated, and the pseudo-labels are estimated.

4.3 Finding the samples with highly accurate pseudo-labels

The classifier needs to be trained by the samples having highly accurate pseudo-labels. To check and select highly accurate pseudo-labels, we utilize the ratio of positive and negative polarity scores obtained from a review to determine the confidence score. If the review r consists of n sentences, $s_1, s_2, s_3, \ldots, s_n$ with positive polarity scores of $P_{pos}(s_1)$, $P_{pos}(s_2), \ldots, P_{pos}(s_n)$, and negative polarity scores of $P_{neg}(s_1)$, $P_{neg}(s_2), \ldots, P_{neg}(s_n)$, then overall positive polarity score of review r is calculated as $P_{pos}(r) = \sum_{i=1}^{n} P_{pos}(s_i)$. Negative polarity score is calculated as $P_{neg}(r) = \sum_{i=1}^{n} P_{neg}(s_i)$. The confidence score of the review r is determined by:

$$ConfScore = \frac{abs\big(P_{pos}(r) + P_{neg}(r)\big)}{abs(P_{pos}(r)) + abs(P_{neg}(r))} \tag{17}$$

In each review, we calculate the mean confidence score (mcs) and standard deviation (std) across all the predictions to find the threshold thr value. The thr value determines various confidence groups, which is calculated as $thr = mcs + std$. The confidence group of review r, confGroup(r) is determined as follows,

$$ConfGroup(r) = \begin{cases} high & if \quad ConfScore \geq thr \\ low & if \quad ConfScore < thr \end{cases} \tag{18}$$

The predicted reviews with a confidence score above the thr fall into the high confidence group. The next category (low) contains predictions with confidence scores below the thr value. Three criteria are considered while categorizing predictions into two groups described.

a) Minimize the inclusion of wrong prediction (i.e., highly accurate pseudo-label) into a group so that it can be used as training data for the classifier with minimal error propagation.
b) Maximize the number of reviews (more extensive training set) utilized as pseudo-labels for the classifier.
c) Show the correlation between the confidence score and the accuracy (i.e., a high confidence score implies high accuracy).

(a) and (b) both are important for having good performance from machine learning and deep learning classifiers, as (a) highly-accurate pseudo-label means less error-propagation to the classifier and (b) a higher number of pseudo-labels means the more extensive training set, that is needed to have good accuracy from machine learning model. (c) is important for group selection, (c) determines which groups should be used as training data and which ones to use as testing data. We find that discretizing the reviews' predictions into two categories best fulfills the above criteria. After identifying highly confident predictions (high confidence groups), we utilize them as pseudo-labeled training data for the classifier.

4.4 CNN-GRU classifier

This research uses a combination of CNN and GRU for sentiment classification. As explained in Section 2.3, it has been proved that CNNs can improve text classification accuracy since they have a strong capacity for extracting local and deep features from text using convolutional layers [56]. On the other hand, GRUs, explained in Section 2.4, can learn the long-term dependencies, so they are appropriate for modeling sequential data such as text because the sentences can be considered a sequence of words from left to right. GRU networks offer less computational complexity and simpler architecture than LSTM. Considering these facts and inspired by the results of [56] proving that the CNN and GRU achieved higher accuracy in text classification tasks, we use a combination of CNN and GRU for sentiment classification. Additionally, we use the Word2vec [24] embedding method to convert the words into dense feature vectors. As shown in Fig. 3, the proposed CNN-GRU architecture includes the embedding, convolution, max-pooling, GRU, and fully connected layers:

a) Embedding layer: This layer receives the labeled samples as word embeddings. Assume v is the vocabulary size of the corpus, and d is the size of word embedding (dimension size). Then, an embedding matrix $EM \in R^{d*v}$ containing all words of the vocabulary is created. Subsequently, a sentence and its embedding can be represented as (19) and (20), respectively:

$$Sentence = [w_1, w_2, \ldots, w_l] \tag{19}$$

$$Sentence\_Embedding = [we_1, we_2, \ldots, we_l], \quad Sentence\_Embedding \in R^{d*l} \tag{20}$$

Where w_i indicates the $i-th$ word of the sentence, l is the length of the sentence and the column we_i denotes the word embedding of w_i, $we_i = EM[w_i]$, $we_i \in R^d$.

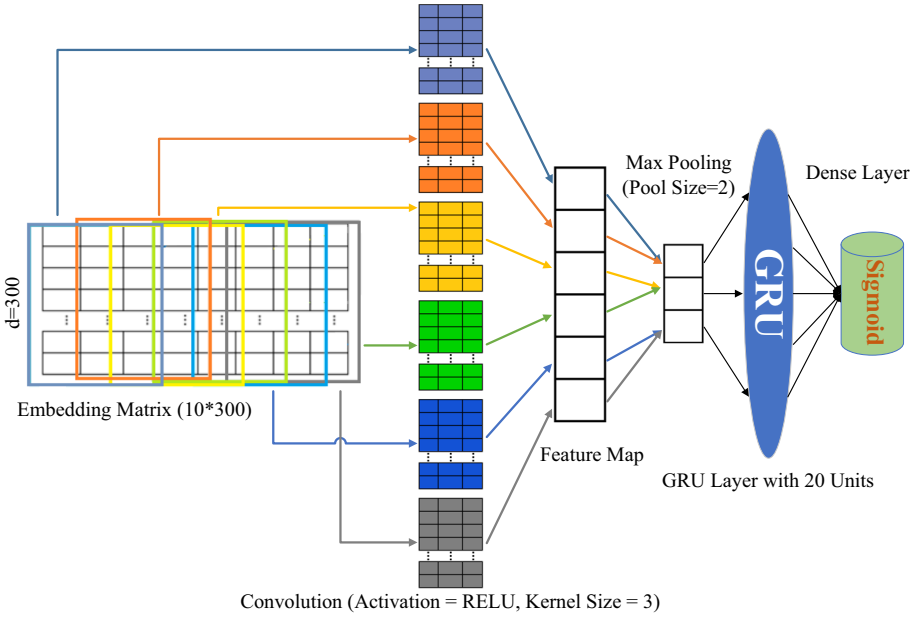

Fig. 3 Architecture of CNN-GRU Classifier

b) Convolution layer: This layer extracts the local features. Suppose $K \in R^{d*w}$ is the kernel size, which is applied to each window of size w, a bias term is added to the result of the convolutional operation, and a feature map $fm \in R^{l-w+1}$ is created as follows:

$$FM = [fm_1, fm_2, \ldots, fm_(l-w+1)], \quad fm \in R^{l-w+1} \tag{21}$$

Then, the following equation shows the $i-th$ element of the feature map:

$$fm_i = \sigma(\sum(EM[*, i:i+w] \circ K) + b \tag{22}$$

Where σ is a non-linear activation function like ReLu or tanh.

c) Pooling layer: In the next step, the feature maps are fed into the pooling layer to find the essential features and reduce the dimensions. The pooling layer, widely used after the CNN layers, performs dimension reduction and consequently decreases the computation time. For example, we use the Max-pooling layer with a pool size equal to 2, which converts the feature map of size $l-w+1$ to $\lfloor \frac{l-w+1}{2} \rfloor$. The output of the pooling layer is:

$$P = \left[p_1, p_2, \ldots, p_{\lfloor \frac{l-w+1}{2} \rfloor}\right], p \in R^{\lfloor \frac{l-w+1}{2} \rfloor} \tag{23}$$

Where p_i is calculated as follows:

$$p_i = \max(fm_{2*i-1}, fm_{2*i}) \tag{24}$$

d) GRU layer: The GRU layer receives the features obtained by the pooling layer to find the long-term dependencies. The output of GRU is $g \in R^n$, encoding a complete sentence.

e) Fully connected layer: The output of the GRU layer is sent to a fully connected layer that uses the $sigmoid$ activation function. Passing the feature vectors to the $sigmoid$ function yields a probability score over sentiment classes. $Sigmoid$ function is calculated as follows:

$$sigmoid(g) = \frac{1}{(1+e^{-g})} \tag{25}$$

Where g denotes the advanced feature vector created by the GRU.

5 Evaluation

During this section, first, we describe the proposed method's evaluation setup and metrics, datasets, and hyperparameter settings. Later on, we compare the results of the proposed method with the other lexicons, base-line classifiers, and similar methods. Finally, we calculate the computational complexity of the proposed method.

5.1 Evaluation setup and metrics

We used the Google Colab platform, with a K80 GPU and 12 GB of RAM, to run the proposed method. In addition, the proposed method was implemented by the Python Language version 3.8 and the Keras library [57] to implement the DNNs. Since the DNNs use random initialization, they give different results in each run. We ran each algorithm ten times and reported the average results.

Since the purpose of the proposed method is to predict the label of texts as positive and negative, the number of true and false predicted labels plays an important role in evaluating

it. Hence, we utilized Accuracy, Precision, Recall, and F1-Score, which are widely used to evaluate the classification tasks. Additionally, similar existing TSA approaches utilized these metrics for evaluating their methods, so we can compare our method with them [28].

Accuracy, which is the number of correct choices relative to all choices, is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

Where:

- TP: the number of samples where the predicted class label and the actual class label are positive.
- FP: the number of samples where the predicted class label is positive, but the actual class label is negative.
- FN: the number of samples where the predicted class label is negative, but the actual class is positive.
- TN: the number of samples where the predicted class label is negative and the actual class label is negative.

Precision calculates the number of class labels truly predicted for each class. This metric is calculated using Eq 21:

$$Precision = \frac{TP}{TP + FP} \tag{27}$$

The recall metric is the weighted average of the correct labels which are correctly predicted for each class and calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{28}$$

F1-score is the harmonic mean of precision and recall metrics.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{29}$$

5.2 Dataset description

To show that the proposed method is independent of domains, we have chosen five English review datasets of different domains, such as movies, books, DVDs, electronics, and kitchens. The first dataset is the second version of the Pang Lee dataset [37], known as Movie Review (MR02), containing 1000 positive and 1000 negative reviews on movies collected from IMDB [9]. What is more, we have employed the Multi-Domain Dataset (MDS), collected by Blitzer et al. [34], containing the reviews of four different domains (Book, DVD, Electronics, and Kitchen) from Amazon [10]. Table 2 shows the number of positive samples (#ps), the number of negative samples (#ns), the number of positive words (#pw), the number of negative words (#nw), and the number of negations (#neg) corresponding each dataset.

5.3 Hyperparameter setting

To set the hyperparameters of the proposed method, we performed it with different values of hyperparameters and found the best values. We set the vector size to make dense feature vectors as 100, 200, and 300. The vector size of 200 achieved the lowest error rate.

[9] https://www.imdb.com/

[10] https://www.amazon.com/

Table 2 Description of datasets

Dataset	#ps	#ns	#pw	#nw	#neg
MR02 Dataset	1000	1000	44,335	46,140	1,027
Book Dataset	1000	1000	35,601	30,283	1,946
DVD Dataset	1000	1000	24,946	11,843	662
Electronic Dataset	1000	1000	24,807	16,633	1,836
Kitchen Dataset	1000	1000	23,939	15,391	1,779

(#ps = the number of positive samples, #ns = the number of negative samples, #pw = the number of positive words, #nw = the number of negative words, #neg = the number of negations)

Also, the Word2vec works based on the skip-gram and Continuous Bag-of-Word (CBOW). We tested each of them, and the skip-gram obtained the lowest error.

In addition, we used the Doc2vec model to convert the samples into document vectors. As proposed in [23], the Doc2vec works in PV-DM and PV-DBOW modes. Like the continuous bag of words, the former is more complex but performs better. The PV-DM method either concatenates or averages all word embeddings of a document to calculate document embeddings. Like skip-gram, the latter is simpler and usually leads to a higher error rate. As a result, we made three Doc2vec models: D2V-DM-Concat, D2V-DM-Average, and D2V-DBOW. We made a logistic regression model on each dataset to compare these models. Its input is document embeddings, and its target is the sentiment labels. Subsequently, Lee and Mikolov suggest that the concatenation of document embeddings created by PV-DM and PV-DBOW improves the performance of the Doc2Vec model [23]. To this end, we concatenated two distributed models (D2V-DM-Concat and D2V-DM-Average) with D2V-DBOW separately and created two concatenated models named D2V-BOW-Concat and D2V-BOW-Average. The error rates of the two concatenated models decreased remarkably, and the D2V-BOW-Concat achieved the lowest error. Finally, we used this model in the following steps. Moreover, we configured the CNN-GRU classifier as listed in Table 3.

5.4 Comparison with lexicons and other classifiers

First, this section compares the proposed method with other widely used lexicons such as TextBlob, SentiStrength, AFINN, VADER, and Flair. Later on, we compare it with other classifiers. To compare with the lexicons, we used the APIs presented by these lexicons

Table 3 Hyperparameters OF CNN-GRU classifier

Hyperparameter	Value
Optimizer	Adam
Loss	Binary Cross-Entropy
Learning Rate	0.01
Stride	1
Activation (Convolution Layer)	ReLu
Activation (GRU Layer)	ReLu
Activation (Dense Layer)	Sigmoid
Epochs	100
Batch size	32
Test Split	0.2

to classify the sentiments of datasets. TextBlob [50] provides an API for some NLP tasks such as SA, part-of-speech tagging, and noun phrase extraction. SentiStrength [58] employs word-matching tools to classify the text and outputs a number demonstrating the polarity of the text. AFINN [46] lexicon generates a number between -5 and +5 to show the sentiment of texts. VADER (Valence Aware Dictionary and sEntiment Reasoner) [51] is a rule-based SA tool. Flair [59] employs an embedding method called contextualized string embedding. As shown in Table 4, Flair, which uses contextual embedding, obtained the closest results to the proposed method, but still, the proposed method outperforms others.

Now, the proposed method is compared with other classifiers. As listed in Table 5, the results obtained by CNN-LSTM are close to the proposed method, and in the case of the DVD dataset, the accuracy of CNN-LSTM (0.83) is higher than the proposed method (0.75). In other cases, the proposed method outperforms the other classifiers.

Table 4 Comparison of the proposed method with other lexicons

Datasets	Methods	Acc	Pre	Rec	F1
MR02	TextBlob	0.59	0.72	0.59	0.65
	SentiStrength	0.58	0.58	0.58	0.58
	AFINN	0.66	0.68	0.67	0.67
	VADER	0.63	0.66	0.63	0.64
	Flair	0.82	0.85	0.82	0.83
	Proposed	**0.84**	0.85	0.82	0.83
Book	TextBlob	0.62	0.72	0.62	0.67
	SentiStrength	0.63	0.65	0.64	0.64
	AFINN	0.65	0.68	0.65	0.66
	VADER	0.63	0.69	0.63	0.66
	Flair	0.75	0.81	0.75	0.78
	Proposed	**0.77**	0.82	0.81	0.81
DVD	TextBlob	0.67	0.75	0.67	0.71
	SentiStrength	0.64	0.66	0.64	0.65
	AFINN	0.68	0.72	0.68	0.70
	VADER	0.65	0.70	0.65	0.67
	Flair	0.72	0.79	0.72	0.75
	Proposed	**0.75**	0.84	0.85	0.85
Electronic	TextBlob	0.65	0.74	0.65	0.69
	SentiStrength	0.70	0.73	0.70	0.71
	AFINN	0.70	0.73	0.70	0.71
	VADER	0.70	0.75	0.70	0.72
	Flair	0.73	0.80	0.73	0.76
	Proposed	**0.83**	0.85	0.84	0.84
Kitchen	TextBlob	0.66	0.76	0.66	0.71
	SentiStrength	0.68	0.73	0.68	0.07
	AFINN	0.73	0.78	0.74	0.76
	VADER	0.70	0.78	0.70	0.74
	Flair	0.81	0.84	0.82	0.83
	Proposed	**0.82**	0.83	0.84	0.83

Table 5 Comparison of the proposed method with other classifiers

Dataset	Model	Acc	Pre	Rec	F1
MR02	CNN	0.81	0.76	0.79	0.77
	GRU	0.80	0.72	0.76	0.74
	LSTM	0.77	0.71	0.76	0.73
	CNN-LSTM	**0.83**	0.78	0.79	0.78
	Proposed(CNN-GRU)	**0.84**	0.85	0.82	**0.83**
Book	CNN	0.74	0.76	0.79	0.77
	GRU	0.75	0.72	0.76	0.74
	LSTM	0.75	0.71	0.76	0.73
	CNN-LSTM	**0.76**	0.78	0.79	0.78
	Proposed(CNN-GRU)	**0.77**	0.82	0.81	**0.81**
DVD	CNN	0.74	0.76	0.79	0.77
	GRU	0.73	0.72	0.76	0.74
	LSTM	0.74	0.71	0.76	0.73
	CNN-LSTM	**0.83**	0.78	0.79	0.78
	Proposed(CNN-GRU)	**0.75**	0.84	0.85	**0.85**
Electronic	CNN	0.81	0.76	0.79	0.77
	GRU	0.80	0.72	0.76	0.74
	LSTM	0.77	0.71	0.76	0.73
	CNN-LSTM	**0.82**	0.78	0.79	0.78
	Proposed(CNN-GRU)	**0.83**	0.85	0.84	**0.84**
Kitchen	CNN	0.81	0.76	0.79	0.77
	GRU	0.80	0.72	0.76	0.74
	LSTM	0.77	0.71	0.76	0.73
	CNN-LSTM	**0.81**	0.78	0.79	0.78
	Proposed(CNN-GRU)	**0.82**	0.83	0.84	**0.83**

5.5 Comparison with other methods

In this section, we compare the proposed method with unsupervised methods such as Zhou et al. [38], Fernandez et al. [43], and Vilares et al. [44], and self-supervised methods such as He and Zhou [36], SSentiA [9], and SESS [33], explained in Section 3.1. These methods employed the MR02 and MDS datasets, the same as the proposed method. As can be seen in Table 6, the results obtained by the proposed method are better than the results reported by the authors of other methods, only in the case of the Kitchen dataset, SESS [33] achieved the higher F1-score.

As explained in Section 3.1, Zhang et al. [33], He and Zhou [36], Fernandez et al. [43], and Sazzed et al. [9] borrow a list of seeds from domain-independent lexicons to calculate the sentiment score of each document. Later on, they train a classifier. Finally, they employ this trained classifier to predict the label of unseen data. Mostly they aggregate the sentiment score of the words forming a document to calculate its sentiment score. In comparison with the proposed method, these methods do not consider the semantic relationships between the documents and their sentiment words using the Soft-Cosine measure.

Zhou et al. [38] uses a method called graph co-regularized non-negative matrix to find the label of documents. However, this method only uses the Cosine similarity to find the similarity

Table 6 Comparison of the proposed method with other similar methods

Dataset	Method	Desc	Acc	Prec	Rec	F1
MR02	He and Zhou	Self-supervised	0.75	-	-	-
	Zhou et al.	Graph Co-Regularization	0.74	-	-	-
	Fernandez et al.	Unsupervised	0.75	0.75	0.75	0.75
	Vilares et al.	Unsupervised	0.74	-	-	-
	SSentiA (LR)	Self-supervised	0.75	0.75	0.75	0.75
	SSentiA (SVM)	Self-supervised	0.77	0.78	0.77	0.77
	Proposed method	Self-supervised	**0.84**	**0.85**	0.82	0.83
Book	SESS	Self-supervised	-	-	-	0.79
	He and Zhou	Self-supervised	0.70	-	-	-
	Proposed method	Self-supervised	0.77	0.82	0.81	0.81
DVD	SESS	Self-supervised	-	-	-	0.80
	He and Zhou	Self-supervised	0.74	-	-	-
	Proposed method	Self-supervised	0.75	0.84	**0.85**	**0.85**
Electronic	SESS	Self-supervised	-	-	-	0.83
	He and Zhou	Self-supervised	0.80	-	-	-
	Proposed method	Self-supervised	0.83	**0.85**	0.84	0.84
Kitchen	SESS	Self-supervised	-	-	-	**0.85**
	He and Zhou	Self-supervised	0.76	-	-	-
	Proposed method	Self-supervised	0.82	0.83	0.84	0.83

between the documents and sentiment words, which is not enough, as we explained in Section 2.2, so its results are not close to the proposed method.

On the other hand, these methods usually use the TF-IDF to convert the texts into feature vectors. Against contextual embedding methods such as Doc2Vec, TF-IDF does not consider the contexts and semantic meaning of texts and will be slow for larger vocabularies. As a result, these methods are not scalable and cannot be utilized for larger datasets with larger vocabularies. The proposed method does not have this limitation. The proposed method can be used in different domains as we evaluated it on the datasets of various domains

Additionally, these methods train classifiers like SVM or LR that do not capture long-term dependencies while processing sequential data like texts. In comparison, as explained in Section 2.3, the proposed method utilizes the CNN-GRU classifier that finds the local and deep features using CNN and captures long-term dependencies using GRU.

5.6 Complexity complexity

Regarding the computational complexity, the complexity of the proposed method is non-trivial and depends on the building blocks of the method, including Doc2vec, the Soft-Cosine, and the CNN-GRU classifier. The complexity of the Doc2vec embedding method is linear since it is composed of a single-layer model. So, it is presented by $O(N)$, where N is the number of documents. The complexity of Soft-Cosine is at most $O(L \times N)$, in which each document length equals L.

Additionally, the complexity of the CNN and GRU are $O(s \times n \times d^2)$ and $O(n \times d^2)$ where s, n, and d are kernel size, sequence length, and representation dimension, respectively [60]. So, the complexity of CNN-GRU is calculated as below:

$$O(CNNGRU) = O(s \times n \times d^2) + O(n \times d^2) \tag{30}$$

Finally, the complexity of the proposed method is:

$$O(ProposedMethod) = O(N) + O(N \times L) + O(s \times n \times d^2) + O(n \times d^2) \tag{31}$$

In real-world datasets, N is much bigger than L, so L can be omitted in (31). Indeed, the number of samples in the dataset (N) is always much higher than the length of samples (L). Moreover, the s is constant. Therefore, the complexity of the proposed method can be calculated by the below equation:

$$O(ProposedMethod) = O(N) + O(n \times d^2) \tag{32}$$

6 Conclusion and future work

SA is a domain-dependent task, so the knowledge-based SA methods that use domain-independent lexicons can not recognize the sentiment of domain-dependent words, and the ML methods trained on a specific domain can not be utilized in other domains. To address this problem, this research proposes an SA method that considers the domain of samples using contextual embeddings, Soft-Cosine similarity, and a CNN-GRU classifier. The proposed method offers a semantic-based pseudo-label generator that estimates the pseudo-labels based on the Soft-Cosine similarity and the number of sentiment words. It uses a list of positive and negative seeds to extract the sentiment words of each sample.

In addition, since the classifier needs to be trained by samples having highly accurate pseudo-labels, another method based on the confidence score is proposed to find the highly accurate pseudo-labels. Then, the samples having highly accurate pseudo-labels are fed into a hybrid CNN-GRU classifier. The CNNs can extract local features deeply, and GRUs capture long-term dependencies. The evaluation results demonstrate that the proposed method outperforms the existing similar approaches.

The comparison of semantic-based pseudo-label generator with other similar existing SA methods such as TextBlob, SentiStrength, VADER, AFINN, and Flair demonstrates that using contextual embeddings and semantic similarity jointly can solve the problem of SA methods, not considering the domain of domain-dependent sentiment words while extracting their sentiments. Contextual embedding methods such as Doc2Vec convert the text into dense feature vectors, and the Soft-Cosine similarity calculates the semantic similarity between the feature vectors of samples and their corresponding sentiment words. Later on, these similarities are used to estimate pseudo-labels.

Additionally, the comparison of CNN-GRU with other classifiers such as CNN, LSTM, GRU, and CNN-LSTM shows that the proposed CNN-GRU classifier outperforms other classifiers in terms of accuracy, precision, recall, and F1. Just in the case of the DVD dataset, the CNN-LSTM performs better than the CNN-GRU.

In the future, the transformers could enhance the proposed method to create more meaningful feature vectors. The Transformers such as RoBERTa and ALBERT were trained on the huge amount of data like Wikipedia and Books and utilized the attention mechanism to overcome the vanishing gradient problem. As a result, they create rich feature vectors that consist of semantic aspects of texts.

In addition, the other text similarity measures, such as Jaccard, can be compared with the Cosine and Soft-Cosine measures, and the proposed method can be extended to estimate the pseudo-labels as a range of numbers to express the intensity of sentiments. Moreover, the proposed method can be evaluated on the datasets of other languages.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability The data that support the findings of this study are publicly available.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Noorian Avval AA, Harounabadi A (2023) A hybrid recommender system using topic modeling and prefixspan algorithm in social media. Complex & Intelligent Systems 9(4):4457–4482. https://doi.org/10.1007/s40747-022-00958-5
2. Al-Smadi M, Qawasmeh O, Al-Ayyoub M, Jararweh Y, Gupta B (2018) Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of arabic hotels' reviews. Journal of computational science 27:386–393. https://doi.org/10.1016/j.jocs.2017.11.006
3. Padminivalli VSJRK, Rao MVPCS, Narne NSR (2023) Sentiment based emotion classification in unstructured textual data using dual stage deep model. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-023-16314-9
4. Seilsepour A, Ravanmehr R, Sima HR (2019) 2016 olympic games on twitter: Sentiment analysis of sports fans tweets using big data framework. Journal of Advances in Computer Engineering and Technology 5(3):143–160
5. Seilsepour A, Alizadeh M, Ravanmehr R, Beheshti MT, Nassiri R (2022) Self-supervised sentiment classification based on semantic similarity measures and contextual embedding using metaheuristic optimizer. In: 2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), pp. 1–7. https://doi.org/10.1109/ICSPIS56952.2022.10043914 . IEEE
6. Mohamad Sham N, Mohamed A (2022) Climate change sentiment analysis using lexicon, machine learning and hybrid approaches. Sustainability 14(8). https://doi.org/10.3390/su14084723
7. Panahandeh Nigjeh M, Ghanbari S (2023) Leveraging parsbert for cross-domain polarity sentiment classification of persian social media comments. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-023-16067-5
8. Iqbal F, Hashmi JM, Fung BCM, Batool R, Khattak AM, Aleem S, Hung PCK (2019) A hybrid framework for sentiment analysis using genetic algorithm based feature reduction. IEEE Access 7:14637–14652. https://doi.org/10.1109/ACCESS.2019.2892852
9. Sazzed S, Jayarathna S (2021) Ssentia: a self-supervised sentiment analyzer for classification from unlabeled data. Machine Learning with Applications 4:100026. https://doi.org/10.1016/j.mlwa.2021.100026
10. Aljedaani W, Rustam F, Mkaouer MW, Ghallab A, Rupapara V, Washington PB, Lee E, Ashraf I (2022) Sentiment analysis on twitter data integrating textblob and deep learning models: The case of us airline industry. Knowl-Based Syst 255:109780. https://doi.org/10.1016/j.knosys.2022.109780
11. Mardjo A, Choksuchat C (2022) Hyvadrf: Hybrid vader–random forest and gwo for bitcoin tweet sentiment analysis. IEEE Access 10:101889–101897. https://doi.org/10.1109/ACCESS.2022.3209662

12. Mohamed A, Zain ZM, Shaiba H, Alturki N, Aldehim G, Sakri S, Yatin SF, Zain JM (2023) Lexdeep: Hybrid lexicon and deep learning sentiment analysis using twitter for unemployment-related discussions during covid-19. Computers, Materials & Continua 75(1):1577–1601. https://doi.org/10.32604/cmc.2023.034746
13. Khan J, Ahmad N, Khalid S, Ali F, Lee Y (2023) Sentiment and context-aware hybrid dnn with attention for text sentiment classification. IEEE Access 11:28162–28179. https://doi.org/10.1109/ACCESS.2023.3259107
14. Kathuria A, Gupta A, Singla R (2023) Aoh-senti: Aspect-oriented hybrid approach to sentiment analysis of students' feedback. SN Computer Science 4(2):152. https://doi.org/10.1007/s42979-022-01611-1
15. Seilsepour A, Ravanmehr R, Nassiri R (2023) Sstsa: A self-supervised topic sentiment analysis using semantic similarity measures and transformers. International Journal of Information Technology & Decision Making 1–39. https://doi.org/10.1142/S0219622023500736
16. Xie W, Fu X, Zhang X, Lu Y, Wei Y, Yang J (2019) Topic sentiment analysis using words embeddings dependency in edge social system. Transactions on Emerging Telecommunications Technologies 3817. https://doi.org/10.1002/ett.3817
17. Seilsepour A, Ravanmehr R, Nassiri R (2023) Topic sentiment analysis based on deep neural network using document embedding technique. The Journal of Supercomputing 1–39
18. Rendón-Cardona P, Gil-Gonzalez J, Páez-Valdez J, Rivera-Henao M (2022) Self-supervised sentiment analysis in spanish to understand the understand the university narrative of the colombian conflict. Appl Sci 12(11):5472. https://doi.org/10.3390/app12115472
19. Sidorov G, Gelbukh A, Gómez-Adorno H, Pinto D (2014) Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas 18(3):491–504
20. Wang J, Dong Y (2020) Measurement of text similarity: a survey. Information 11(9):421. https://doi.org/10.3390/info11090421
21. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
22. Zulqarnain M, Abd Ishak S, Ghazali R, Nawi NM, Aamir M, Hassim YMM (2020) An improved deep learning approach based on variant two-state gated recurrent unit and word embeddings for sentiment classification. International Journal of Advanced Computer Science and Applications 11(1). https://doi.org/10.14569/IJACSA.2020.0110174
23. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196. https://doi.org/10.5555/3044805.3045025. PMLR
24. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. CoRR
25. Li B, Han L (2013) Distance weighted cosine similarity measure for text classification. In: Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China. Proceedings 14, pp. 611–618. Springer
26. Ravanmehr R, Mohamadrezaei R (2024) Deep Learning Overview, pp. 27–72. Springer, Cham. https://doi.org/10.1007/978-3-031-42559-2_2
27. Gupta B, Prakasam P, Velmurugan T (2022) Integrated bert embeddings, bilstmbigru and 1-d cnn model for binary sentiment classification analysis of movie reviews. Multimedia Tools and Applications 81(23):33067–33086. https://doi.org/10.1007/s11042-022-13155-w
28. Alizadeh M, Mousavi SE, Beheshti MT, Ostadi A (2021) Combination of feature selection and hybrid classifier as to network intrusion detection system adopting fa, gwo, and bat optimizers. In: 2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS), pp. 1–7. https://doi.org/10.1109/ICSPIS54653.2021.9729365. IEEE
29. Cheruku R, Hussain K, Kavati I, Reddy AM, Reddy KS (2023) Sentiment classification with modified roberta and recurrent neural networks. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-023-16833-5
30. Alizadeh M, Beheshti MT, Ramezani A, Saadatinezhad H (2020) Network traffic forecasting based on fixed telecommunication data using deep learning. In: 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), pp. 1–7. https://doi.org/10.1109/ICSPIS51611.2020.9349573. IEEE
31. Alizadeh M, Beheshti MT, Ramezani A, Bolouki S (2023) An optimized hybrid methodology for short-term traffic forecasting in telecommunication networks. Transactions on Emerging Telecommunications Technologies 34(12):4860
32. Noorian A, Harounabadi A, Hazratifard M (2023) A sequential neural recommendation system exploiting bert and lstm on social media posts. Complex & Intelligent Systems. https://doi.org/10.1007/s40747-023-01191-4

33. Zhang W, Zhao K, Qiu L, Hu C (2009) Sess: A self-supervised and syntax-based method for sentiment classification. In: Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2, pp. 596–605. https://doi.org/10.1016/j.mlwa.2021.100026
34. Blitzer J, Dredze M, Pereira F (2007) Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440–447
35. Qiu L, Zhang W, Hu C, Zhao K (2009) Selc: a self-supervised model for sentiment classification. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 929–936. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1645953.1646072
36. He Y, Zhou D (2011) Self-training from labeled features for sentiment analysis. Information Processing & Management 47(4):606–616. https://doi.org/10.1016/j.ipm.2010.11.003
37. Pang B, Lee L (2004) A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. Association for Computational Linguistics. doi 10(3115/1218955):1218990
38. Zhou G, Zhao J, Zeng D (2014) Sentiment classification with graph co-regularization. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 1331–1340. Dublin City University and Association for Computational Linguistics, Dublin, Ireland
39. Muhammad A, Wiratunga N, Lothian R (2016) Contextual sentiment analysis for social media genres. Knowl-Based Syst 108:92–101. https://doi.org/10.1016/j.knosys.2016.05.032
40. Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A (2010) Sentiment strength detection in short informal text. J Am Soc Inform Sci Technol 61(12):2544–2558. https://doi.org/10.1002/asi.21416
41. Jiméénez-Zafra SM, Martín-Valdivia MT, Martínez-Cámara E, Ureña-López LA (2016) Combining resources to improve unsupervised sentiment analysis at aspect level. J Inf Sci 42(2):213–229. https://doi.org/10.1177/0165551515593686
42. Baccianella S, Esuli A, Sebastiani F et al (2010) Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. Lrec 10:2200–2204
43. Fernández-Gavilanes M, Álvarez-López T, Juncal-Martínez J, Costa-Montenegro E, González-Castaño FJ (2016) Unsupervised method for sentiment analysis in online texts. Expert Syst Appl 58:57–75. https://doi.org/10.1016/j.eswa.2016.03.031
44. Vilares D, Gómez-Rodríguez C, Alonso MA (2017) Universal, unsupervised (rule-based), uncovered sentiment analysis. Knowl-Based Syst 118:45–55. https://doi.org/10.1016/j.knosys.2016.11.014
45. Vashishtha S, Susan S (2020) Fuzzy interpretation of word polarity scores for unsupervised sentiment analysis. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–6. https://doi.org/10.1109/ICCCNT49239.2020.9225646
46. Nielsen FÅ (2011) A new anew: Evaluation of a word list for sentiment analysis in microblogs. arXiv:1103.2903
47. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. HLT'11, pp. 142–150. Association for Computational Linguistics, USA
48. Kusner M, Sun Y, Kolkin N, Weinberger K (2015) From word embeddings to document distances. In: Bach F, Blei D (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 957–966. PMLR, Lille, France
49. Zhang L, Ghosh R, Dekhil M, Hsu M, Liu B (2011) Combining lexicon-based and learning-based methods for twitter sentiment analysis. HP Laboratories, Technical Report HPL-2011 89:1–8. https://doi.org/10.1145/2346676.2346681
50. Loria S, et al (2018) textblob documentation. Release 0.15 2(8)
51. Hutto C, Gilbert E (2014) Vader: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the International AAAI Conference on Web and Social Media 8:216–225. https://doi.org/10.1609/icwsm.v8i1.14550
52. Zhang P, He Z (2013) A weakly supervised approach to chinese sentiment classification using partitioned self-training. J Inf Sci 39(6):815–831. https://doi.org/10.1177/0165551513480330
53. Hu M, Liu B (2004) Mining and summarizing customer reviews. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD'04, pp. 168–177. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1014052.1014073
54. Brooke J, Tofiloski M, Taboada M (2009) Cross-linguistic sentiment analysis: From english to spanish. In: Proceedings of the International Conference RANLP-2009, pp. 50–54
55. Hatzivassiloglou V, McKeown K (1997) Predicting the semantic orientation of adjectives. In: 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, pp. 174–181. Association for Computational Linguistics, USA. https://doi.org/10.3115/976909.979640
56. Wang X, Jiang W, Luo Z (2016) Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: Proceedings of COLING 2016, the 26th International Conference on Computa-

tional Linguistics: Technical Papers, pp. 2428–2437. The COLING 2016 Organizing Committee, Osaka, Japan
57. Chollet F, et al (2015) Keras: Deep learning library for theano and tensorflow. https://keras.io/k 7(8):1
58. Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A (2010) Sentiment strength detection in short informal text. J Am Soc Inform Sci Technol 61(12):2544–2558. https://doi.org/10.1002/asi.21416
59. Akbik A, Blythe D, Vollgraf R (2018) Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, pp. 1638–1649
60. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008