International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT21717

Improving Accuracy of The Sentence-Level Lexicon-Based Sentiment Analysis Using Machine Learning

Titya Eng[1], Md Rashed Ibn Nawab[*2], Kazi Md Shahiduzzaman[3]
[1]University of Battambang, Battambang, Cambodia
[*2]Northwestern Polytechnical University, Xi'an, China
[3]Jatiya Kabi Kazi Nazrul Islam University

ABSTRACT

Sentiment Analysis studies people's attitudes, opinions, evaluations, emotions, sentiments toward some entities such as products, topics, individuals, services, issues and classify them whether the opinion or evaluations inclines to that entities or not. It is getting more research focus in recent years due to its benefits for scientific and commercial purposes. This research aims at developing a better approach for sentiment analysis at the sentence level by using a combination of lexicon resources and a machine learning method. Moreover, as reviews data on the internet is unstructured and has much noise, this research uses different preprocessing techniques to clean the data before processing in different algorithms discussed in subsequent sections. Additionally, the lexicon building processes, how the lexicon is handled and combined with the machine learning algorithm for predicting sentiment is also discussed. In sentiment analysis, sentence's sentiment can be classified into three classes: positive sentiment, negative sentiment, or neutral. However, in this research work, we have excluded neutral sentiment for avoiding ambiguity and unnecessary complexity. The experiment results show that the proposed algorithm outperforms compared to the baseline machine learning algorithms. We have used four distinct datasets and different performance measures to check and validate the proposed method's robustness.

Keywords: Sentiment Analysis; Machine Learning; Support vector machine; Lexicon, Natural language processing.

I. INTRODUCTION

Sentiment analysis, also known as opinion mining, is often modeled as a classification problem. Thus, in this paper, we will use sentiment analysis or sentiment classification interchangeably. Generally, sentiment analysis extracts different features from structured or unstructured textual data and analyzes them to get opinion, emotions, feeling out of it. In this era of the internet, it is relatively easy to get customer or stakeholder voice via different channels, e.g. blog, social media, customer care service, online form and many more. However, an organization can adequately utilize this data when they can retrieve these feedback's emotion or feelings, and at this point, the necessity of the sentiment analysis emerges. Moreover, sentiment analysis relieves us from the manual labelling or annotating the massive amount of data available on the internet, which contains valuable information for business decision making. Interestingly, sentiment analysis has vast application areas, e.g. sales performance prediction, election result prediction, box office revenue prediction, stock market prediction, expert investor identification, trading strategy formulation, characterizing social relations, etc.

Usually, the sentiment classification can be dug down up to three different levels, (i) Document-level classification, (ii) Sentence-level classification, and the more sophisticated (iii) Aspect-level classification. In our experiment, we have conducted a sentence-level sentiment classification. In many reviews, people express more than one opinion in a single product or service review, usually distributed in different sentences. For example, "This mobile phone looks excellent. But it is costly!!". There are two different opinions in this case. These cases strongly advocate the necessity of sentence-level sentiment analysis.

Understanding human sentiment towards different products or services enables better service, a better recommendation system. It also provides crucial insights of market trends. As this data is massive, manually analyzing the data is almost impossible. In this scenario machine learning has come into place to solve this problem. Many researchers have been trying to find the most accurate way for sentiment analysis using different supervised, unsupervised or semi-supervised machine learning algorithms, tuning the different parameters or modifying existing algorithms. As the data related to human language has a complex and ambiguous structure, the machine learning approaches' performance is still insufficient to model this accurately, which reasonably creates a scientific research scope. Moreover, supervised ML approaches have a dependency on voluminous labelled data.

Contrarily, lexicon-based approaches are also popular in sentiment analysis which considers semantic orientation of words in a text and calculates sentiment. A dictionary of positive and negative words is developed in this approach where each positive or negative word is assigned a sentiment value. These values are applied to the review text, converted into a bag of words and mapped with the dictionaries before [1]. Next, a combining function predicts the sentiment of that text. This approach is easy to understand. However, this is a slower process as it requires matching among massive data. Besides, the lexicon building process also plays a key role.

In this context, we need to find a way to handle the complexity of the data and provide a faster result with better accuracy than the previous standalone machine learning or lexicon-based methods. This research work addresses three critical issues of sentiment analysis

- Some words have a bipolar meaning in different contexts, e.g. cheap.
- The complexity of articulating linguistic patterns using ML approaches
- Performance optimization of lexicon-based approach

II. LITERATURE REVIEW

Sentiment classification is related to natural language processing, and it is also a part of big data and data mining. There was insufficient research before the beginning of the 20th century in both natural language processing and in linguistics. The lack of opinion text available in digital form was one of the reasons. However, with the advancement of scientific research and the internet and the growing necessity for marketing, analyzing user feedback, and subsequent business decision making, this research area is getting more interest.

Since sentiment classification is a binary classification task, any existing supervised or unsupervised machine learning method can be used, e.g., Naïve Bayes classification, support vector machine (SVM), Maximum Entropy, etc. In [2], we can find the first application of ML algorithms in sentiment analysis, where the researchers classified movie reviews as positive or negative. The author used unigrams (bag of words) as features to feed into machine learning to train and classify the document. This method performed better on both Naïve Bayes and SVM algorithm. The authors in [3] proposed a hybrid model combining KNN and SVM with different preprocessing and feature generation techniques and showed that their model performed better than the standalone ML algorithm. We can find another case of utilizing the KNN and Naïve Bayes algorithm on movie and hotel review datasets in [4]. However, their achieved accuracy level in most cases is below 70%. Another machine learning-based approach using the NLTK library on Yelp dataset is proposed in [5]. Following their approach, they managed to achieve 79.12% accuracy using the Naïve Bayes algorithm.

On the contrary, sometimes researchers in sentiment analysis also focus on reducing the dependency on the ML technique because of its limitations in capturing complex linguistic structure and nonlocal contextual cues. Being inspired by this issue, authors in [6] proposed a sentence-level, context-aware approach which is capable of modelling both local and global contextual information. Improving CRF models, this approach performs better than the state-of-the-art supervised and unsupervised methods.

A different semantic approach using a lexical resource like SentiWordNet is proposed by Cernian et al., where results predicted by the system are compared against the star ratings from the Amazon dataset [7]. A 61% average success rate is attained while validating against 300 product reviews, which is relatively low compared to our accuracy level.

We can find another lexicon-based approach where the authors proposed a general-purpose WKWSCI Sentiment Lexicon [8]. This lexicon performs similar to other state-of-the-art lexicons in product review categorization. However, it performed the best in sentiment analysis of news headlines with 69% accuracy.

As emoticon is a critical feature of review data while performing sentiment analysis, Wang et al. examined the connection between emoticon and opinion in Twitter data. In this experiment, he compared the performance with and without considering emoticon. This study concludes that promising results can be achieved in sentiment analysis by careful dealing with emoticons [9].

III. METHODOLOGY

A. Proposed Architecture Overview

Our research proposed a hybrid method combining lexical resources and machine learning for sentence-level sentiment classification on online reviews. Fig. 1 shows the system architecture of our proposed method where it is divisible into two main sections, (1) lexicon resources, and (2) application of SVM (Support Vector Machine), a machine learning algorithm.

Our experiment's lexical resources include sentiment, negation, intensifier, emoticon, adjective sense disambiguation, phrase and idiom, syntax pattern, and exception. These lexicon resources work together with the machine learning algorithm to calculate a given sentence's sentiment score. Like other classification tasks, our dataset is divided into two parts, training dataset, and testing dataset. However, before feeding the data into the classifier, we need to execute the following steps,

- Data Preprocessing: tokenization, remove stop words and punctuation, cleansing symbols, and links. After preprocessing, the Bag of Word model is constructed.
- Transformation: As the machine learning algorithms only accept numbers, we need to convert text into numbers. Here we have used the TF-IDF technique for this transformation. Now the input data has become Bag of Features. After this transformation, we fed the Bag of Features into the SVM classifier to train it.

- Before feeding the testing dataset, we again need to perform data preprocessing. In this step, we have performed uppercase to lowercase conversion and removed stop words and punctuation.
- Next, we have checked the sentences with the phrase and idiom lexicon and replaced them with a score from the lexicon.
- After that, emoticons in the sentences are replaced with a score from the lexicon.
- In this step, we have tokenized the sentences into a Bag of Word.
- Finally, we have calculated the sentiment score using lexicon resources and combined it with the machine learning score to predict the final result.

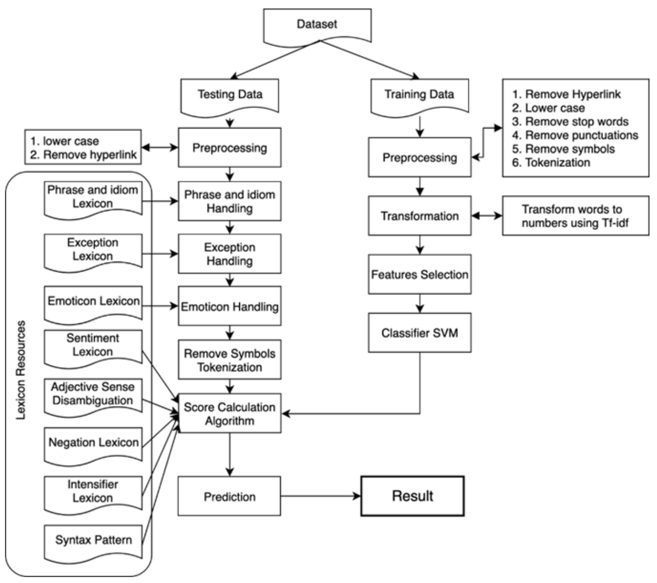

Figure 1. System Architecture of Proposed Method

B. Dataset

Considering all the constraints, we will use the JSON format dataset on the Yelp website [10] in our experiment. The dataset contains many reviews from different domains encompassing restaurants, hotels, food, shopping, bar, beauty salon, dentists, etc. Firstly, we will collect 10,000 records of sentence-level reviews from the Yelp dataset in the restaurant domain. As collecting datasets is challenging and time-consuming, we employed an algorithm to automatically pick up the sentiment sentences from the reviews and automatically annotate the label using star ratings in the dataset. Here, we picked the last sentence of the review as it is most likely to contain sentiment and label it according to the star rating given by reviews. We annotated the two-star and one-star reviews as negative sentiment sentences and set the sentiment score to 0. Four-star and five-star reviews are marked as positive sentiment sentences. Hence, we set the sentiment score for these sentences to 1. We also considered that three-star reviews bear no sentiment, thus omitted. Encompassing all these considerations, Algorithm-1 filters the sentiment sentences to form a suitable dataset for lexicon preparation and testing.

Algorithm-1: Filter Review Sentence from Dataset

```
   Input: $N \leftarrow Yelp\ JSON\ Dataset$
   Output: $a\ file\ F\ contain\ list\ of\ [\ sentence\ , label\ ]$
1  $T\ an\ emty\ array\ for\ storing\ result$
2  $\boldsymbol{for\ each}\ item\ \in\ N\ \boldsymbol{do}$
3          $R\ \leftarrow item\ [\ 'text'\ ]$
4          $>quick\ check\ if\ R\ contains\ restaurant\ keywords$
5          $\mathrm{ST} \leftarrow\ sent\_tokenize(\ R\ )\ [-1]$
6          $SRS\ \leftarrow\ 1$
7          $\boldsymbol{if}\ item\ [\ 'stars'\ ]\ ==\ 3\ \mathbf{then}$
8                  $\boldsymbol{continue}$
9          $\boldsymbol{if}\ item\ [\ 'stars'\ ]\ <\ 3\ \boldsymbol{then}$
10                 $SRS\ \leftarrow\ 0$
11         $T\ \leftarrow\ [\ R\ ,\ SRS\ ]$
12 $\boldsymbol{end\ for}$
13 $Write\ list\ T\ to\ output\ file\ F$
```

TABLE I. SUMMARY OF THE DATASETS

Dataset	Positive Sentence	Negative Sentence	Total
Dataset 1 [10]	7,408	2,592	10,000
Dataset 2 [11]	923	1,320	2,243
Dataset 3 [12]	1,500	1,500	3,000
Dataset 4 [13]	5,331	5,331	10,662

To validate the performance and to check the robustness of our proposed method against the baseline sentiment classification algorithms (naïve Bayes, support vector machine, random forest, k-nearest neighbor, maximum entropy, Vader Sentiment, and TextBlob), along with the Yelp dataset, we are going to use three other datasets as described in Table I.

C. Lexicon Generation Algorithms and Scoring

(i) Sentiment Lexicon

Sentiment words are an essential part of the lexicon-based analysis. Though we use many sentiment words in our daily life, reviewers use many different, new, and strange sentiment words on the internet. Hence, to achieve better accuracy, we should not just consider commonly used words. In this experiment, we have used the available resources from [13] and the Yelp dataset to build a sentiment lexicon. As the dataset is huge and searching sentiment words manually is a tedious job, we have exploited Algorithm-2 to automate the process.

Algorithm-2: Finding Sentiment Word

```
Input: $N \leftarrow Yelp\ JSON\ Dataset$
Output: $a\ file\ F\ contain\ list\ of\ line\ with\ structure\ S = (R, SSW, SRS)$
T an emty array for storing result
for each item ∈ N do
    R ← item['text']
    if R contains no word from NG:
        SRS ← 1
        if item['stars'] == 3 then
            continue
        if item['stars'] < 3 then
            SRS ← 0
        T ← [R, SRS]
end for
L an empty array for storing result
for each r, srs ∈ T do
    T ← word_tokenize(r)
    T ← remove_stop_words(T)
    T ← remove_symbols(T)
    T ← remove_number(T)
    for each index, w ∈ T do
        if srs > 0 then
            SWS ← SentiWordnet_Score(w)
            if SWS.pos_score > 0.4 then
                L ← [w, 1, #]
            if SWS.neg_score > 0.4 then
                L ← [w, SWS.neg_score, 1]
        else if srs == 0 then
            SWS ← SentiWordnet_Score(w)
            if SWS.pos_score > 0.4 then
                L ← [w, SWS.pos_score, −1]
            if SWS.neg_score > 0.4 then
                L ← [w, 0, #]
    end for
end for
T ← remove duplicate item from L
Write list L to output file F
```

After getting the Yelp dataset as input, Algorithm-2 filters out reviews that have negative words. Then it labels the review texts according to the star rating. If the review item has a four or five stars rating, then the algorithm labels it as positive. For one or two stars, the review item is labeled as negative. The review item that has a three-star rating is omitted as it might contain no sentiment. Next, the result is saved with structure S = (R, SRS) to list T. For every item in list T, the algorithm tokenizes the sentences using the natural language processing toolkit (NLPTK), and then it filters out unnecessary words such as negative words, stop words, numbers, and symbols. After that, for every word in the tokenized sentence, if the label of that sentence is positive, check the score S of the word using the SentiWordnet lexicon resource. If the positive score of S is bigger than 0.4, append S=(W, 1, #) to list L or if the negative score of S is bigger than 0.4, append S=(W, SSW-, 1) to list L. Contrarily, in case of the tokenized sentence labeled as negative, if the positive score of S is bigger than 0.4 then append S=(W, SSW+, -1) to list L or if the negative score of S is bigger than 0.4 then append S=(W, 0, #) to list L. Finally, all duplicate words are removed from list L, and an output file is generated. The sentiment word lexicon is also stored in an SQLite3 database with structure (id, word, value, comment) for further and faster retrieval in our experiment. Next, for score calculation of any review sentence, we need to utilize SQL command and check word by word in the sentiment lexicon. If found, the algorithm retrieves the associated score. The algorithm proceeds searching otherwise.

(ii) Intensifier Lexicon

Intensifier words are adverbs that can influence the strength of the sentiment word. It can lower or higher the strength of the sentiment words or even reverse the value to negative or positive, e.g., very, so, amazingly, tremendously, extremely, totally, amazingly, etc., are intensifiers that strengthen the value of sentiment. The intensifiers such as slightly, almost, could, might, hardly, rarely, seldom, fairly, nearly, etc., lower the strength of the sentiment words. There are also some words that can reverse the sentiment, e.g., less, few, etc. We have used Algorithm-3 and Yelp dataset to build an intensifier lexicon in our experiment.

Algorithm-3: Find Intensifier Words in Dataset

```
Input: $N \leftarrow$ Yelp JSON Dataset
Output: a file F contain list of intensifier
T an emty array for storing result
for each item $\in N$ do
    $R \leftarrow item['text']$
    TK $\leftarrow$ word_tokenize ( R )
    P $\leftarrow$ pos_tag ( TK )
    for each word , pos $\in P$ do
        if word at index k is a sentiment word && it is an adjective then
            if $k > 1$ && word at index $[k-1]$ is an adverb then
                word at index $[k-1]$ is likely to be an intensifier word
                $T \leftarrow$ word at index $[k-1]$
    end for
end for
$T \leftarrow$ remove duplicate item from T
Write to output file F
```

In Algorithm-3, after tokenizing the review text item, if any tokenized word is an adjective and resembles a sentiment word, the word before it is likely to be an intensifier. Besides, if there exists a word before sentiment word and that word is an adverb, the word also needs to enlist. Finally, after removing all duplicate words from the list, an output file containing the list of intensifier words is generated. The final list is checked manually to remove the wrong words if there is any, and a score is assigned to each intensifier according to its meaning. The score lies between [-0.1, 2].

To calculate the intensifier score, firstly, we look for sentiment words in the sentence. If found, need to search for the intensifier to the left for the window size of five. After finding an intensifier, we need to multiply the sentiment word score with the intensifier word score. The searching continues otherwise. The score calculation process is as follows,

*S = S(I)*S(SW)*
*good => very good = (2) * (1) = 2 => positive sentiment*
*bad => so bad = (2) * (-1) = -2 => negative sentiment*
*amazing => absolutely amazing = (2) * (1) = 2 => positive sentiment*
*good => totally good = (2) * (1) = 2 => positive sentiment*

(iii) Negation Lexicon

Negation lexicon resource is a list of negative indicating words. Handling negation is a crucial step in our research work, as it can reverse the polarity or reduce the strength of the sentiment word. Improper management of the negation lexicon may also lead to lower sentiment prediction accuracy. In our experiment, we divided the negation lexicon into three categories, (1) Negative words which change the polarity of the sentiment, e.g. "not good", "lack of good point", "nothing is good", "need to improve", "none of this work", "hasn't any good", "missing good part", "won't work", etc. (2) Negative words which lower the strength of the sentiment when occurred before some sentiment words. It is a list of words that does not invert the meaning of the sentiment, but it changes the intensifier of the opinion words; for example, "not the best" means it's okay, "not perfect" means it is okay, "not excellent" also means it's okay. (3) Negative words with no effect on the sentiment when used before some specific words, e.g. "not only", "no wonder", "no end of", "not to mention", "no matter what", etc.

Negation score is calculated after intensifier scoring. Next, the algorithm searches for a negative word that precedes the sentiment word for a window of five. If found, the sentiment word score is multiplied with the intensifier score. Besides, if the intensifier meets with superlative in sentiment word, then it will not change the polarity of the sentiment word; it only changes the strength to lower value. Score calculation for negation, intensifier, and sentiment word is as follows,

*S = S(N) * S(I) * S(SW)*
*very good => not very good = (-1) (2) * (1) = -2 => negative sentiment*
*bad => not bad = (-1) *(-1) = 1 => positive sentiment*
*the best => not the best => (1/2) * (2) = 1 => still positive sentiment*
*the worst => not the worst => (1/2) * (-1) = 1 => still negative sentiment*

(iv) Emoticon Lexicon

In recent times, we can see the huge popularity and use of different emoticons on social networks, online stores, and review websites like Amazon, Yelp, eBay, etc. Emoticons are also a vital feature in sentiment analysis. Here, we have utilized the compiled list of positive and negative emoticons used in [9]. We also took the effort to manually add more emoticons to the list. After that, each emoticon is labelled with a score depending on the polarity of the sentiment. A positive emoticon has a +2 score, and a negative emoticon has -2 score. If an emoticon is found in a sentence, it is replaced by # and assign its score to the algorithm for further calculation. Importantly, the score for the emoticon needs to calculate before the symbol preprocessing step. The algorithm first gets each of the pair of (emoticon, value) in the lexicon and search for emoticon in the review sentence. If found, then it removes that emoticon and assigns the score to the review sentence.

(v) Phrase and Idiom Lexicon

Phrase and idiom is another critical lexicon to deal with in our research. Here we have considered the phrases and idioms which trigger sentiment, e.g., phrases like "stay away", "hardworking", "mess up", "over price", "can't stand", "let down", "make fun of" etc. and idioms like "chew someone out", "cost someone an arm and a leg", "shooting fish in a barrel", "a piece of cake", and many more. As phrases and idioms are formed by more than one word, so it must be handled after text preprocessing and before tokenization. For dealing with this issue, if any phrase and idiom are found in a sentence, our proposed method would replace those words with a word, "good" or "bad", depending on the meaning of the phrase and idiom. So it can be further processed by our negation and intensifier algorithm for score calculation. For example,

over price *=> bad =>* ***negative*** *sentiment,*
not ***over price*** *=> not bad =>* ***positive*** *sentiment*
can't stand *=> bad =>* ***negative*** *sentiment*
thumb up *=> good =>* ***positive*** *sentiment*
break a leg *(means good luck) => good =>* ***positive*** *sentiment*

There are also some group of words which contain negation word, but it does not change the meaning of that phrase at all. We have addressed these phrases as the exception lexicon. This lexicon is stored together with the phrases and idioms lexicon file because we will process them at the same time with a very similar algorithm before the tokenization step and obviously after text preprocessing. The only difference is it will not add any score to the total score of the sentence. To do so, we need to check each item in the exception lexicon, and if found, we need to simply remove the phrase from the sentence, e.g.

not only => remove from sentence
no wonder => remove from sentence
not to mention => remove from sentence
not just => remove from sentence

(vi) Adjective Sense Disambiguation

Adjective sense disambiguation is a technique used to differentiate the meaning of an adjective in a different context. For example, cheap ticket, cheap flight, reveals positive sentiment. Contrarily, cheap seller, cheap quality, cheap material triggers negative sentiment towards the seller. In our proposed method, we explored the adjective sense disambiguation using a nearby noun.

Algorithm-4: Adjective Sense Disambiguation

```
Input:   N ← Yelp JSON Dataset ,   AW ← list of ambiguous adjectives
                 NG ← list of negative words
Output: a file F contain list of line with structure S = ( NN , SW , SV , CS )
T an emty array for storing result
for each item ∈ N do
        R ← item['text']
        if R contains no word from NG:
                SRS ← 1
                if item['stars'] == 3 then
                        continue
                if item['stars'] < 3 then
                        SRS ← 0
                T ← [ R , SRS ]
end for
for each word ∈ AW do
        LW an emty array for storing result
        for each r , srs ∈ T do
                T ← word_tokenize (r)
                T ← remove_stop_words (T)
                T ← remove_symbols (T)
                T ← remove_number (T)
                P ← pos_tag(T)
                for each index , w , pos ∈ P do
                        if w == word then
                                for i in range ( index + 1 , index + 3 ) do
                                        > quick check if P[ w ][ i ] is an English word
                                        > quick check if P[ pos ][ i ] is a noun
                                        > quick check if P[ w ][i ] is not in LW
                                        LW ← [ SRS , w , P[ w ][i ], r]
                                end for
                                for i in range( index − 1 , index − 5 ,−1 ) do
                                        > quick check if P[ w ][i ]is verb tobe
                                        for j in range ( i − 1 , i − 3 , −1 ) do
                                                > quick check if P[ w ][ i ] is an English word
                                                > quick check if P[ pos ][ i ] is a noun
                                                > quick check if P[ w ][i ] is not in LW
                                                LW ← [ SRS , w , P[ w ][i ] , r]
                                end for
                        end if
                end for
        end for
        create a file F name 'word.csv'
        F ← LW
end for
```

Algorithm-4 and Yelp dataset is used to query nearby nouns of sentiment words and label the sentiment of that co-occurrence. In this task, we have only considered the most commonly used and bipolar adjective only, e.g., great, hot, cheap, crazy, small, big, high, low, heavy, light, huge, thick, deep, and few, etc.

Algorithm-5: Score Calculation Algorithm

```
     Input:     $N \leftarrow$ processed sentence from previous step
                $S \leftarrow$ score from previous step processed with emoticon lexicon
     Output: total score calculate by the algorithm
1    $T \leftarrow$ word_tokenize ( $N$ )
2    Total $\leftarrow S$
3    for each index , item $\in N$ do
4        $s \leftarrow 0$
5        if $N$[ index ] is a sentiment word:
6            replace found word with #
7            if $N$[ index ] is ambiguous adjective then. # check sentiment lexicon
8                nlist $\leftarrow$ ambiguous noun list of sentiment word $N$[ index ]
9                if $N$[index + 1] or $N$[ index + 2 ] in nlist then
10                   $s \leftarrow (-1) * sentimentScore( N[ index ] )$
11               else
12                   $s \leftarrow sentimentScore(N[ index ])$
13               replace found word with #
14           for i in range( index − 1 , index − 5 , −1) do
15               if $N$[ i ] is an intensifier word then
16                   $s \leftarrow s * intensifierScore(N[ i ])$ # check in intensifier lexicon
17                   replace found word with #
18                   > quick check if $N$[ i ] is not in punctuation list → go to 33
19                   for j in range( i − 1 , i − 5 , −1 ) do
20                       > quick check if $N$[ i ] is not in punctuation list
21                       if $N$[ i ] is a negative word then # check in negation lexicon
22                           replace found word with #
23                           $s \leftarrow s * negativeScore(N[ i ])$ → go to 33
24                   end for
25               if $N$[ i ] is a negative word then
26                   $s \leftarrow s * negativeScore( N[ i ] )$ # check in negation lexicon
27                   replace found word with #
28                   > quick check if $N$[ i ] is not in punctuation list → go to 33
29                   for j in range( i − 1 , i − 5 , −1 ) do
30                       > quick check if $N$[ i ] is not in punctuation list
31                       if $N$[ i ] is a intensifier word then #check intensifier lexicon
32                           replace found word with #
33                           $s \leftarrow s * intensifierScore( N[ i ] )$ → go to 33
34                   end for
35           end for
36       $T \leftarrow T + s$
37       $s \leftarrow 0$
38   end for
```

After manual inspection to remove wrong words, this file is saved into the SQLite3 database with the following structure S = (Id, Word, Value, IsAmbiguous, Pos_Nouns, Neg_Nouns). Here, Word is the sentiment word. Value is the value of a sentiment word, and it can be [-1, -2, 1, 2]. IsAmbiguous indicates if that sentiment word is ambiguous. Pos_Nouns contains a list of nouns that change the polarity of sentiment words to positive. Neg_Nouns contains a list of nouns that change the polarity of sentiment words to negative.

(vii) Syntax Pattern

Besides all the different types of lexicons, there exists some syntax that looks like a negative sentence, but it is actually a positive sentence, e.g., I can't recommend it enough, I couldn't agree more, I can't recommend this any higher, etc. So, if we follow the steps we just discussed, it may result in an incorrect prediction. For example, "Can't" in "I can't recommend it enough" is an intensifier word that is equal to -1, recommend is a sentiment word that is equal to 1. Hence, "can't recommend" = (-1) * (1) = -1, which is definitely not the real meaning of the sentence. Actually, this is a positive sentiment sentence that means it is so good that the opinion holder cannot describe it enough. These sentences can be handled by looking for sentences that contain sentiment words in the pattern: "couldn't…more" or "can't…more", "…can't recommend…higher", etc., then score them according to the intrinsic meaning.

(viii) Score Calculation Algorithm

Finally, Algorithm-5 is used to accumulate and combine all the calculated scores using the lexicon resources discussed above to get the sentiment prediction score.

D. Machine Learning Model Training Phase

As per our proposed methodology (Fig. 1), we also involved the machine learning algorithm (SVM) with our score calculation algorithm to predict the final result. Fig. 2 depicts the processes before feeding training data into the machine learning algorithm.

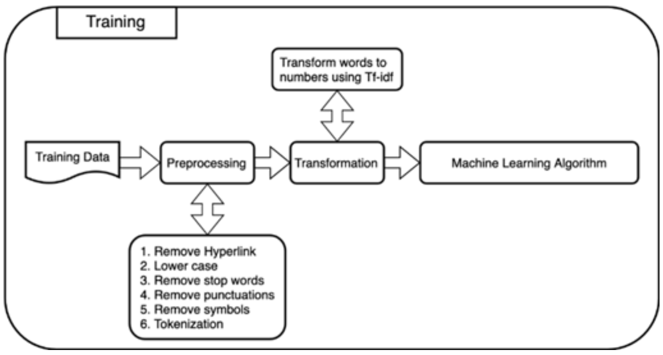

Figure 2. ML Model Training

Here, we have used natural language processing techniques for data preprocessing, including removing hyperlinks, removing all punctuations, removing all stop words, lowercase all words in the sentence, cleaning all symbols, and tokenizing to split the input sentence into a list of words. In the transformation step, the TF-IDF technique is used to transform the tokenized words into a vector. Finally, the processed data with the sentiment label of each sentence is fed into the machine learning classifier.

E. Machine Learning Model Testing Phase

In this section, we will discuss how we get the final score on the testing data. The overall testing phase is portrayed in Fig. 3. The preprocessing in the testing phase is slightly different from the training phase as it requires removing hyperlinks and making lowercase only before the lexicon-based scoring. Next, the data is cross-checked with Phrase and Idiom, Exception, and Emoticon lexicon and processed as discussed in the previous section. After removing symbols and tokenization, the data is fed to the score calculation algorithm, which utilizes different techniques discussed above to handle intensifier, negation, sentiment, adjective sends disambiguation to calculate the total score of the sentences. According to our proposed method, the final sentiment prediction score is calculated using the following expression,

$$S(sent) = \begin{cases} S(sentL) \quad , & for\ S(sentL) \neq 0 \\ S(sentSVM), & for\ S(sentL) = 0 \end{cases}$$

Where: - *S(sentL)*: total score of the sentence using lexicons.
- *S(sent)*: final score of the proposed system.
- *S(sentSVM)*: score by Support Vector Machine.

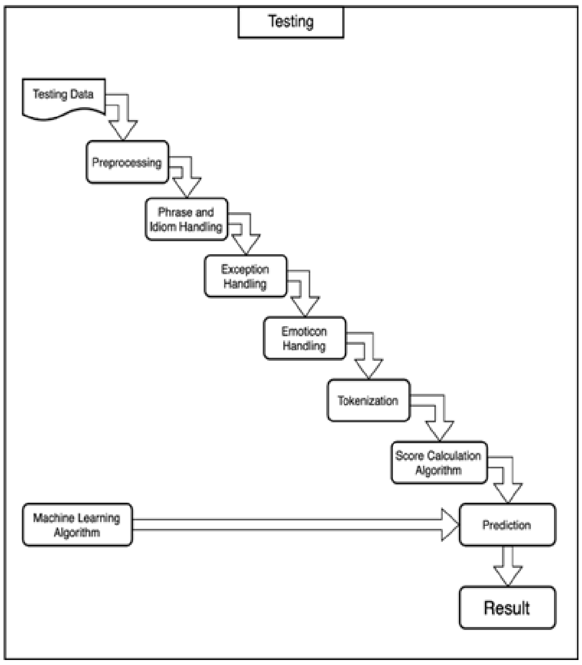

Figure 3. ML Model Testing

F. Performance Optimization

As our proposed method involves a lexicon-based approach with huge text preprocessing job, the system is prone to longer execution time. Nowadays, even personal computer is getting more powerful with multiple cores and hyper-threading technology. As each record of testing data is independent, we can take the advantage to split it into n partitions and share every partition with each core of the CPU. Here, each partition performs the classification task independently using the proposed method. As depicted in Fig. 4, all predictions are combined to get the final score, which brings sufficient improvement in execution time than the single-core utilizing approach.

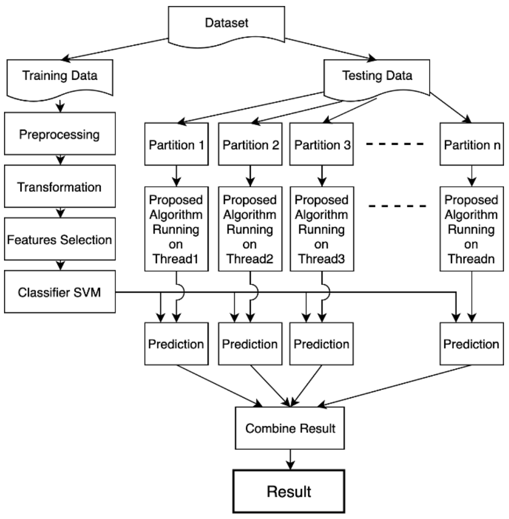

Figure 4. Performance Optimization

G. System Evaluation Methods

To validate the results by our proposed method, we have used different performance measures, e.g., Accuracy, Confusion Matrix, ROC (Receiver Operating Characteristics) Curve, and MCC (Matthews Correlation Coefficient) Score, and Algorithm Efficiency.

IV. RESULT ANALYSIS

Fig. 5 demonstrates that our proposed method performs better than any standalone machine learning classifier like NB, SVM, KNN, etc., or lexicon-based sentiment classifiers like Vader or TextBlob. Although Naïve Bayes and SVM's accuracy performance is closer to our proposed method and relatively stable in all four cases, other algorithms fluctuated on different datasets. Our accuracy claim is further strengthened by the ROC Curve (Fig. 6), Confusion Matrix (Fig. 7), and MCC Score (Table II). An efficiency comparison is also portrayed in Table III, where efficiency is a performance measure that expresses the average execution time an algorithm takes to complete a specific task. Here, because of using lexical resources, our proposed method takes a bit longer training and testing time; utilizing SQLite3 and the parallel computing techniques discussed above, the training time of 6.73s and testing time of 7.20s is attained.

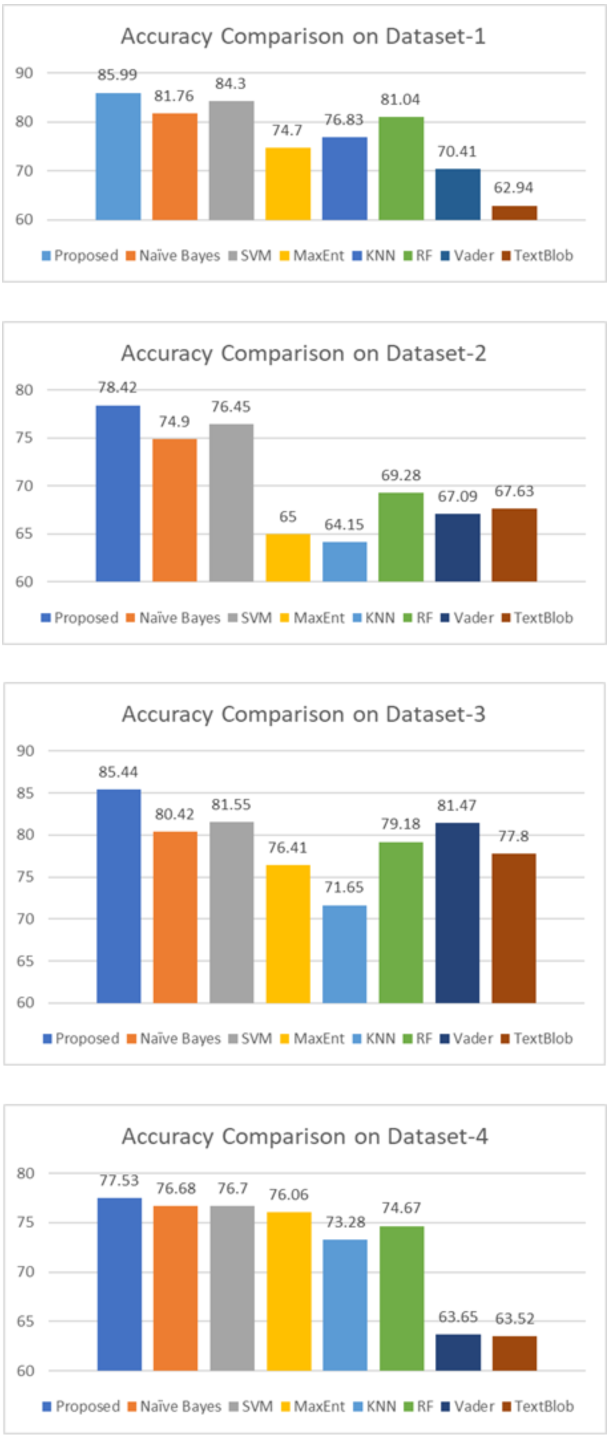

Figure 5. Accuracy Comparison

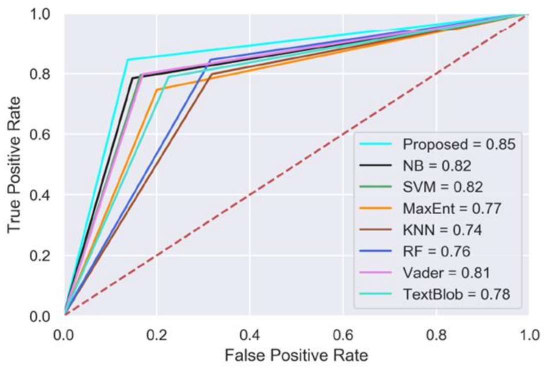

Figure 6. ROC Curve

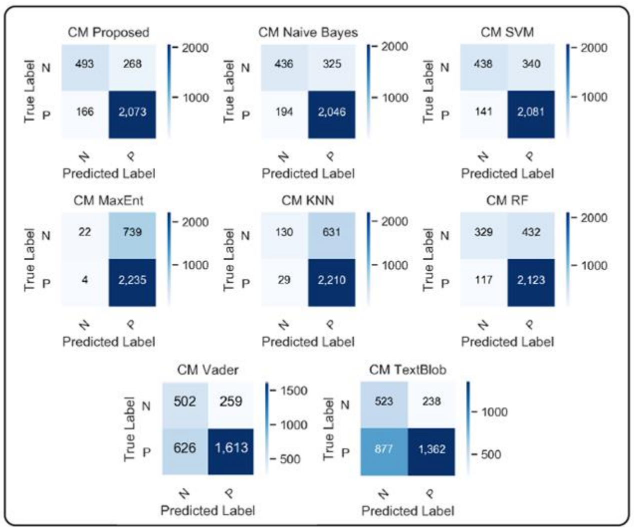

Figure 7. Confusion Matrix

TABLE II. MCC SCORE COMPARISON

Classifier	MCC Score on Dataset1	MCC Score on Dataset2	MCC Score on Dataset3	MCC Score on Dataset4
NB	0.47	0.42	0.66	0.54
SVM	0.55	0.48	0.63	0.55
MaxEnt	0.11	0.21	0.56	0.53
KNN	0.23	0.28	0.49	0.47
Random Forest	0.39	0.37	0.56	0.50
Vader	0.33	0.30	0.62	0.28
TextBlob	0.26	0.35	0.55	0.27
Proposed	**0.60**	**0.54**	**0.73**	**0.58**

TABLE III. EFFICIENCY COMPARISON

Classifier	Training Time	Testing Time
NB	0.12	0.05
SVM	6.73	2.48
MaxEnt	5.06	1.37
KNN	1.19	1.17
Random Forest	10.89	4.46
Vader	N/A	1.40
TextBlob	N/A	0.77
Proposed	6.73	7.20

V. CONCLUSION

Sentiment analysis has become one of the vital techniques in making business decisions as it directly involves the consumer group. Despite the current progress in this research area, there are still many challenges as the human opinion and write up in the form of review is complex and ambiguous. In our research work, we made an effort to combine the lexical resources and machine learning model like SVM to classify sentiment at the sentence-level, and we have got promising results (>85%) compared to popular lexicon-based or standalone machine learning classification algorithms.

VI. REFERENCES

[1] . A. Jurek, M. D. Mulvenna, and Y. Bi, "Improved lexicon-based sentiment analysis for social media analytics," Secur. Inform., vol. 4, no. 1, p. 9, Dec. 2015.

[2] . B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02, Not Known, 2002, vol. 10, pp. 79–86.

[3] . A. Gupta, J. Pruthi, and N. Sahu, "Sentiment Analysis of Tweets using Machine Learning Approach," Int. J. Comput. Sci. Mob. Comput., vol. 6, no. 4, pp. 444–458, Apr. 2017.

[4] . Department of Computer Science & Engineering, Heritage Institute of Technology, Kolkata, India, L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari, "Sentiment Analysis of Review Datasets Using Naïve Bayes' and K-NN Classifier," Int. J. Inf. Eng. Electron. Bus., vol. 8, no. 4, pp. 54–62, Jul. 2016.

[5] . H. S and R. Ramathmika, "Sentiment Analysis of Yelp Reviews by Machine Learning," in

2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 700–704.

[6] . B. Yang and C. Cardie, "Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, 2014, pp. 325–335.

[7] . A. Cernian, V. Sgarciu, and B. Martin, "Sentiment analysis from product reviews using SentiWordNet as lexical resource," in 2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2015, p. WE-15-WE-18.

[8] . C. S. Khoo and S. B. Johnkhan, "Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons," J. Inf. Sci., vol. 44, no. 4, pp. 491–511, Aug. 2018.

[9] . H. Wang and J. A. Castanon, "Sentiment expression via emoticons on social media," in 2015 IEEE International Conference on Big Data (Big Data), 2015, pp. 2404–2408.

[10] . https://www.yelp.com/dataset (accessed Aug. 03, 2020).

[11] . O. Täckström and R. McDonald, "Discovering Fine-Grained Sentiment with Latent Variable Structured Prediction Models," in Proceedings of the 33rd European Conference on Advances in Information Retrieval, Berlin, Heidelberg, 2011, pp. 368–374.

[12] . D. Kotzias, M. Denil, N. de Freitas, and P. Smyth, "From Group to Individual Labels Using Deep Features," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2015, pp. 597–606.

[13] . B. Pang and L. Lee, "Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales," in Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05, Ann Arbor, Michigan, 2005, pp. 115–124.

Cite this Article

Titya Eng, Md Rashed Ibn Nawab, Kazi Md Shahiduzzaman, "Improving Accuracy of The Sentence-Level Lexicon-Based Sentiment Analysis Using Machine Learning", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 1, pp. 57-69, January-February 2021. Available at doi : https://doi.org/10.32628/CSEIT21717
Journal URL : http://ijsrcseit.com/CSEIT21717