Sentiment Analysis Using Out of Core Learning

Mahmudul Hasan, Ishrak Islam, K. M. Azharul Hasan
Department of Computer Science and Engineering
Khulna University of Engineering and Technology, Bangladesh
Email: mahmudul@cse.kuet.ac.bd, ishrak.islam@gmail.com, az@cse.kuet.ac.bd

***Abstract*—Text sentiment detection for a particular language other than English is one of the challenging tasks presently. The reasons are; it needs a large dataset, language has no specific structure, one word has a different meaning, and it is hard for even human to understand the connotation of particular words. There exists several proposed architecture for detecting emotions in the Bengali language using machine learning and deep learning approaches, but they are not accurate enough to predict the perfect emotion of the sentence. And there is still no standalone architecture is available that can extract the sentiments hidden inside of a sentence in different languages. In this paper, we are proposing an abstract model that can enable sentiment analysis without any restriction of using a fixed language somewhat applicable to any language. With the use of natural language processing, we have extracted the features, and these features are then fed to different machine learning models for classification. As our main concern was to build up a general model, this model is confined to binary classification, i.e., positive and negative. Apart from this, In our system architecture, we have implemented stochastic gradient descent for optimization. So our model can be called out of core learning model where the model can be updated when new user data is inserted without training the whole model. For the evaluation of the performance of our model, we have trained the estimators against Bangla translated IMDB review dataset and calculated different evaluation metrics for our estimators. The dataset is translated into Bangla using google translator.**

***Index Terms*—Sentiment Analysis, Logistic regression, Perceptron, Data Preprocessing, Feature Extraction, Tokenizing, Stemming, Tagging, Stemmer, TF-IDF.**

I. INTRODUCTION

Tremendous development in computer science is happening as more decades are passing by. Computers are made more capable and more user-friendly. Besides the enormous amount of information like text, image and video are generated over the internet every day. Classification of ascending data, filtering spam, defining the meaning and finding a pattern of this kind of massive data become a significant issue in present era [1]. For solving these problems and making digital components more humanoid, human languages are needed to be understood by the computer. Now with the help of natural language processing and machine learning algorithm it has become easy for us to convert human texts in more like computer form. One of the applications of this field is called sentiment analysis which involves building a system to collect and categorize human opinions for a particular subject [2], [3]. Also, it predicts future patterns and behaviors, allowing decision makers to make decisions more practically. Automated sentiment analysis often uses machine learning, to mine text for sentiment. Sentiment analysis can be useful in several ways. It is essential for organizations which sell products online because reviews of products have a significant impact on the purchase. Different types of social media are used as a marketing platform nowadays because people feel more comfortable sharing their opinion about products on social media. Moreover, other ones are getting more biased towards this review, comment, and feedback about the products. Hence sentiment analysis has become a new business strategy in this era. Again another thing essential for developing an artificially intelligent agent that it should rely upon the human opinion to make decisions [4], [5] The main application of sentiment classification are in the broad areas on text mining and natural language processing including to get product review, opinion in a specific matter and text classification.

II. RELATED WORK

There have been several approaches taken by the researchers to model human language in machine form [6]–[8]. [9] presented a sentiment detection technique using valancy of a word. But the valacy was calculated using English sentiwordnet. A N-gram based sentiment detection was proposed in [10] for bangla natural text. The sohwed the performance increases for 2-grams when negative words occurs more in a sentence. In [11] the authors used Long Short-Term Memory Network (LSTM) and Gated Recurrent Units (GRU) to classify ticket in the right category in the ticket system. Total classes in this paper were 66 and data size was 217,000. In [12] python, nltk was used for processing text, and Naive Bayes algorithm was applied for classifying text. In [13] they predicted user's sentiment polarity using Lexicon based classifier. They created dataset manually from Flipkart, Snapdeal, Amazon and some review forums. However, they have not taken any machine learning approach. In [14] they gave a review on some paper containing text classification using clustering and principal component analysis (PCA). [15] authors gave an idea about two architectures for online advertising. The proposed architecture is Stand-alone Architecture and Scalable Repository-based Architecture. In [16] they proposed Multinomial Naive Bayes (MNB) classifier combined with Bayesian Networks (BN) classifier incorporated with feature extraction and feature selection. Ten-fold cross-validation model was used for evaluation. They have used review polarity dataset and Reuters-21578 text collection for training and testing. A web based sentiment classification for Bangla text was implemented in

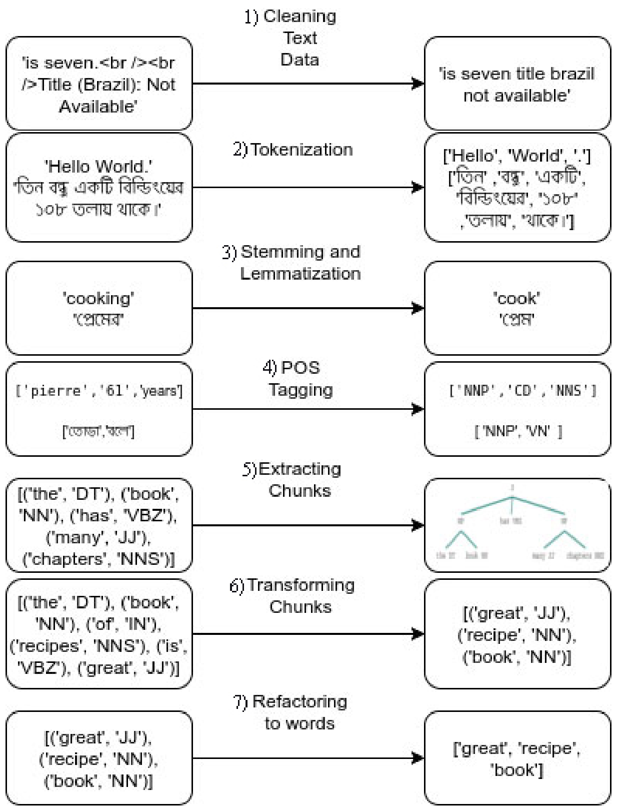

1) Cleaning Text Data
'is seven.<br /><br />Title (Brazil): Not Available'
'is seven title brazil not available'
2) Tokenization
'Hello World.'
'তিন বন্ধু একটি বিল্ডিংয়ের ১০৮ তলায় থাকে।'
['Hello', 'World', '.']
['তিন' ,'বন্ধু', 'একটি', 'বিল্ডিংয়ের', '১০৮' ,'তলায়', 'থাকে।']
3) Stemming and Lemmatization
'cooking'
'প্রেমের'
'cook'
'প্রেম'
4) POS Tagging
['pierre','61','years']
['তোতা','বলে']
['NNP','CD','NNS']
['NNP', 'VN']
5) Extracting Chunks
[('the', 'DT'), ('book', 'NN'), ('has', 'VBZ'), ('many', 'JJ'), ('chapters', 'NNS')]
6) Transforming Chunks
[('the', 'DT'), ('book', 'NN'), ('of', 'IN'), ('recipes', 'NNS'), ('is', 'VBZ'), ('great', 'JJ')]
[('great', 'JJ'), ('recipe', 'NN'), ('book', 'NN')]
7) Refactoring to words
[('great', 'JJ'), ('recipe', 'NN'), ('book', 'NN')]
['great', 'recipe', 'book']

Fig. 1: Implementation of filtering

[17]. The authors presented diverse combination of Bangla words from the web using SVM.

III. PROPOSED FRAMEWORK FOR SENTIMENT ANALYSIS

In this paper, we propose a framework for sentiment mining where the following five phases are sequentially combined for data extraction, preprocessing and building a model.

A. Preprocessing

Before feeding data to estimators, we have to clean the text datasets. From the knowledge of Natural Language Processing, we can filter sentences and get the more significant features. The steps are shown in figure 1:

1) Cleaning all text data: First of all, the text might contain HTML markup as well as punctuation and other non-letter characters. While HTML markup does not contain much helpful semantics but punctuation marks can represent useful, additional information in specific NLP contexts. We used a regular expression like

$$<[^{>}]*>= \text{Detects anytype of Markup Language}$$

and

$$(?::|;|=)(?:-)?(?:)|(|D|P) = \text{Detects}$$
$$\text{Emoticons from text}$$

for finding out HTML markup and emoticons. After that, we can strip out these from the texts.

2) Tokenizing data into sentences: Tokenization of data means that we have to split sentence or phrases into words or tokens. Tokenization is mandatory in preprocessing. Tokenization of a sentence is necessary for filtering essential words [18]. Suppose a sentence ”Quick Brown Fox” can be tokenized into three words ”Quick”, ”Brown” and ”Fox”.

3) Stemming and Lemmatizing words : Stemming is a strategy to expel fastens from a word, winding up with the stem. By far, there are four established stemmers: PorterStemmer, RegexpStemmer, SnowballStemmer, LancasterStemmer [1]. We have utilized PorterStemmer for execution of assessment mining.

Like Stemming, Lemmatization does the same work. It is mainly used to finding out the root word. It truncates the insignificant portion of words. It also finds the valid lemma and finds out if it exists in a dictionary or not. For Lemmatization, We used Princeton university's wordnet database. Figure 1(3) depicts the process of Stemming and Lemmatization.

4) Part of Speech Tagging: There are many types of words in each, and it is required to identify the label of each word. For this purpose part of speech tagging is used. After that part of speech tagging tuples are created. It remains in a specific form where the list is generated using words and tags respectively. Part of speech is mainly eight types. Among them, our primary concern is on the noun, adjective, and verb which is the sentiment or sense of the sentence [1]. Suppose if we apply POS Tagging on Sentence ”John Likes Her” then it will be ”John (Determiner)”, ”Likes (Verb)”, ”Her (Pronoun)”.

5) Extracting Chunks from the sentence: After part of speech tagging there remains many unnecessary and duplicate words. So it should be filtered out. For this reason, extraction of chunks from the sentence is applied. After extraction, the sentence takes a form of a tree of chunks which can be easily transformed and processed [1].

6) Transforming chunks: Transforming of chunks means replacing words where rearranging words and correcting them takes place without changing the meaning. The changes can resemble as follows:

- Swapping verb phrase
- Transforming plural noun into singular
- Swapping Infinitive Phrase
- Eradicating unnecessary words

The chunk transforms are for grammatical correction and rearranging phrases without loss of significance. Figure 1 (5) represents the process.

7) Refactoring to words: After transformation, chunks are refactored to extract a feature from it. Supposing after preprocessing a sentence ”The book of recipes is delicious” is converted to ”('delicious', 'JJ'), ('recipe', 'NN'), ('book', 'NN')”. After refactoring, we will get ”Delicious Recipe Book”.

B. Feature Extraction

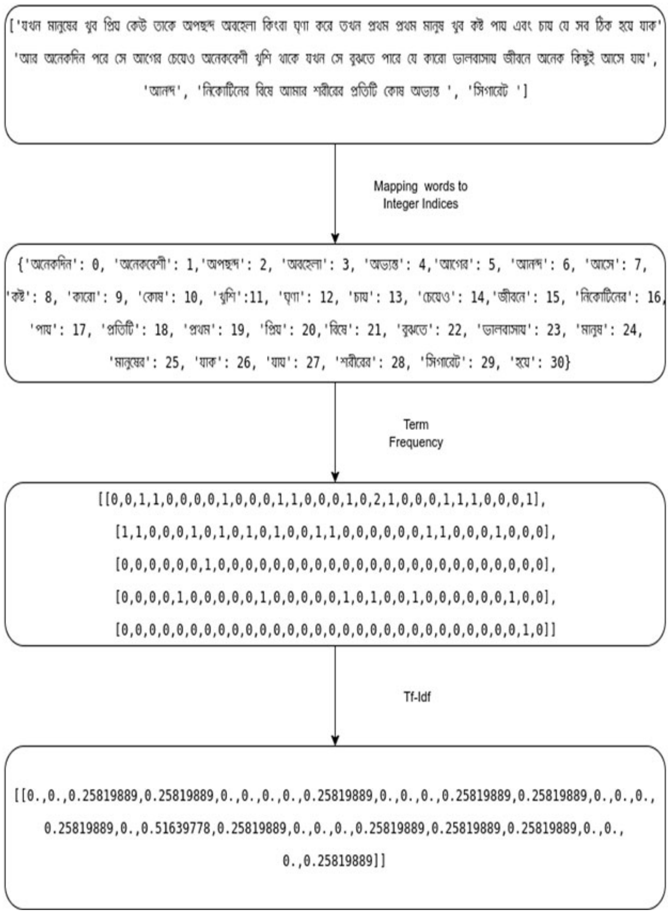
Feature extraction can be done in many ways, but we have used the following two techniques:

1) Term Frequency-Inverse Document Frequency (TF-IDF): Machine learning algorithm needs the data to be represented in a particular form otherwise the algorithms cannot be implemented. So we need to transform the text data into a feature set. In this case, the TF-IDF vectorizer is utilized. It is the weighting factor of the document. It represents how much a word or term is important for the document. Supposing the counted value of the word ”W1” is more than the counted value of the word ”W2” in the document. So the word ”W1” is more important for that document. So the weighting factor is made much bigger in case of ”W1” word. According to [2] the implementation of TF-IDF is following:

tf-idf(x,y) = tf(x,y) * idf(x,y)
where,

- tf-idf(x,y) = TF-IDF value of a term calculation result
- tf(x,y) (term frequency) = Number of same term in document
- idf(x,y) (inverse document frequency) = Calculation of log multiplied by the inverse probability of a term being found in any document

We have shown TF-IDF of a sample of our dataset in Figure 2.



```
['যখন মানুষের খুব প্রিয় কেউ তাকে অপছন্দ অবহেলা কিংবা ঘৃণা করে তখন প্রথম প্রথম মানুষ খুব কষ্ট পায় এবং চায় যে সব ঠিক হয়ে যাক'
'আর অনেকদিন পরে সে আগের চেয়েও অনেকবেশী খুশি থাকে যখন সে বুঝতে পারে যে কারো ভালবাসায় জীবনে অনেক কিছুই আসে যায়',
'আনন্দ', 'নিকোটিনের বিষে আমার শরীরের প্রতিটি কোষ অভ্যস্ত ', 'সিগারেট ']
```

```
{'অনেকদিন': 0, 'অনেকবেশী': 1,'অপছন্দ': 2, 'অবহেলা': 3, 'অভ্যস্ত': 4,'আগের': 5, 'আনন্দ': 6, 'আসে': 7,
'কষ্ট': 8, 'কারো': 9, 'কোষ': 10, 'খুশি':11, 'ঘৃণা': 12, 'চায়': 13, 'চেয়েও': 14,'জীবনে': 15, 'নিকোটিনের': 16,
'পায়': 17, 'প্রতিটি': 18, 'প্রথম': 19, 'প্রিয়': 20,'বিষে': 21, 'বুঝতে': 22, 'ভালবাসায়': 23, 'মানুষ': 24,
'মানুষের': 25, 'যাক': 26, 'যায়': 27, 'শরীরের': 28, 'সিগারেট': 29, 'হয়ে': 30}
```

```
[[0,0,1,1,0,0,0,0,1,0,0,0,1,1,0,0,0,1,0,2,1,0,0,0,1,1,1,0,0,0,1],
 [1,1,0,0,0,1,0,1,0,1,0,1,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0],
 [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,0,1,0,0],
 [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]]
```

```
[[0.,0.,0.25819889,0.25819889,0.,0.,0.,0.,0.25819889,0.,0.,0.,0.25819889,0.25819889,0.,0.,0.,
0.25819889,0.,0.51639778,0.25819889,0.,0.,0.,0.25819889,0.25819889,0.25819889,0.,0.,
0.,0.25819889]]
```

Fig. 2: Word to vector implementation

2) Hashing Vectorizer: When there is a massive amount of data, it is hard to vectorize the whole data. So in this case instead of TF-IDF hashing vectorizer is used. In the case of hashing vectorizer, it does not consider inverse document frequency. Instead, it just hashes each term frequency in such a way that collision is avoided [**?**].

After hashing vectorizer is implemented a token is mapped to feature integer. [2]

Hashing vectorizer has following advantages:

- It is memory efficient. So there is no need to store the whole dictionary of data.
- Hash functions are an efficient way of mapping terms to features, and it does not necessarily need to be applied only to term frequencies.
- As there is no state computed during the fit, it can be utilized as a part of streaming (partial fit) or parallel pipeline.
- It may be possible to reduce the length of the hash feature vector, and so the complexity will also reduce significantly with an acceptable loss of effectiveness or accuracy.
- It is fast and simple.
- Handling of missing data is easy with hashing vectorizer.

There are also a couple of cons (vs. using a CountVectorizer with an in-memory vocabulary):

- There is no inverse mapping.
- Hash collisions may occur at any time.
- There is no IDF weighting.
- A hash table does not accept any null values.
- Based on document frequency hashing vectorizer can not limit features.

C. Training the estimators

For training the estimators, following things must be handled.

1) Developing a system that can read data from different sources like web, hard drive, database, etc.
2) A way of extracting features from these data which is described earlier.
3) An incremental algorithm.

We have used the following classifiers:

1) Perceptron: Rosenblatt’s threshold perceptron model is to use a reductionist approach to mimic how a single neuron in the brain works: it either fires, or it does not [19]. The figure 3 illustrates the general concept of the perceptron:

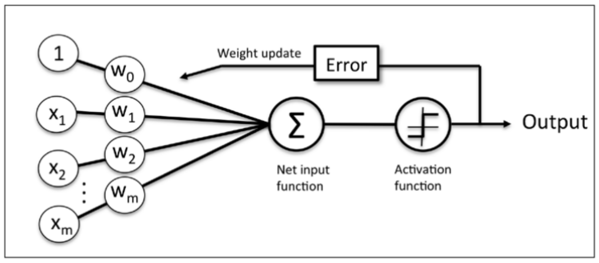


Fig. 3: Perceptron

The equation for a perceptron is as following:

$$\hat{y} = w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n$$

here,
$\hat{y}$ = predicted value
$w_1, w_2, ...w_n$ = weight matrix
$x_1, x_2, ...x_n$ = input matrix
Suppose we are considering two input x1 = 1 and x2 = 0 and weights are $w1 = .8$ and $w2 = .7$, then predicted value would be $.8 * 1 + .7 * 0 = .8$.

2) Logistic Regression: It is similar to the perceptron, but an extra sigmoid function is added here [19]. Figure 4 illustrates the concept of the Logistic Regression:

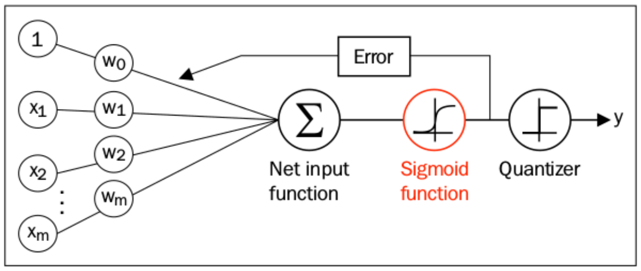


Fig. 4: Logistic Regression Classifier

The equation for sigmoid function is as follows:

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

. So considering the same input that was for perceptron we will get sigmoid value = $\frac{1}{1+e^{-.8}}$ = .342667 .

3) Multinomial Naive Bayes: Naive Bayes is a simple algorithm which is based on the Bayesian theorem. However, Multinomial Naive Bayes is a specialized version of Naive Bayes. They are both supervised learning method [20]. Instead of other distribution, it implements multinomial distribution. Which is mostly related to counting. Hence Multinomial Naive Bayes is more suitable for text processing. Following shows the equation of MNB:

$$P(D|d) = \log \frac{N_c}{N} + \sum_{i=1}^{n} \log \frac{t_i + \alpha}{\sum_{i=1}^{n} t + \alpha}$$

where,
$P(X|c)$ = probability of document D in class d
N_c = total documents in class d
N = total documents
t_i = weighting factor t
$\sum_{i=1}^{n} t$ = total weighting factor in class d
α = smoothing parameter

4) Passive Aggressive Classifier: Passive Aggressive Classifier is one of the best classifiers for online learning [21]. If the model correctly classifies the data, then it is kept, but if it gives the wrong classification, the model is adapted according to the newly entered correct data.

D. Dataset Description

TABLE I: Portion of our Dataset

No.	Review	Sentiment
0	I went and saw this movie last ...	1
1	Actor turned director Bill Paxton ...	1
2	As a recreational golfer with ...	1
3	I saw this film in a sneak preview ...	1
4	Bill Paxton has taken the true story ...	1

Our model is based on IMDB Review dataset described in [22]. The movie review dataset consists of 50,000 polar movie reviews that are labeled as either positive or negative. Table I shows the portion of IMDB Dataset. We have used google translator to translate these reviews into Bengali language and trained our estimators against the translated dataset. There are fifty thousand reviews in the translated dataset which we have split into eighty to twenty ratio for training and testing respectively.

E. Evaluation Criteria

For any classification problems and for sentiment analysis also, following evaluation metrics should be calculated.

1) Accuracy: Accuracy is the measure of how correctly a classifier classifies a data. The equation for the accuracy is as following:

$$\text{Accuracy} = \frac{\text{T.P.} + \text{T.N.}}{\text{T.P.} + \text{T.N.} + \text{F.P.} + \text{F.N.}}$$

Where, T.P. = True Positive, T.N. = True Negative, F.P. = False Positive and F.N. = False Negative.

2) Precision: Precision depicts that how much precisely a classifier can classify the data even if it is a wrong class. But with the help of both accuracy and precision calculation we can estimate classifier's performance. For a single class the precision value is given in the following equation.

$$\text{Precision} = \frac{\text{T.P.}}{\text{T.P.} + \text{F.P.}}$$

3) Recall: Recall means that from the number of samples of one class how many samples are correctly classified as that class. The equation for the recall is following:

$$\text{Recall} = \frac{\text{T.P.}}{\text{T.P.} + \text{F.N.}}$$

4) F1 Score: F1 score is the weighted average of Precision and Recall. Equation of the F1 score is following:

$$\text{F1 Score} = \frac{2 * \text{T.P.}}{2 * \text{T.P.} + \text{F.P.} + \text{F.N.}}$$

IV. RESULT

Figure 5 shows the comparison of accuracy between estimators. In this case, the estimators are trained and tested against IMDB's binary labeled translated database [23], [24]. After building the model, our goal is to take user responses and train the estimators based on them and give a user a dynamic model. We have used web framework Flask from

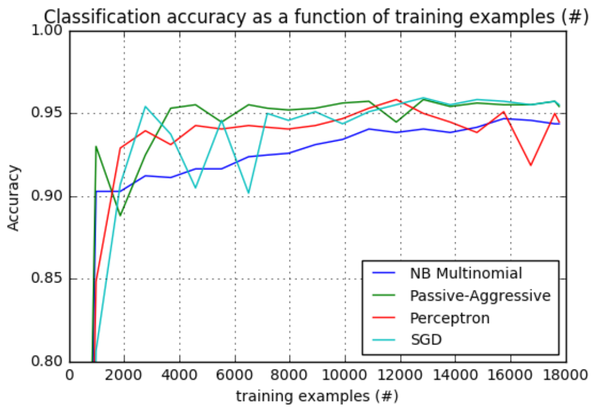


Fig. 5: Evaluation

python and SQLite Database query for implementation of our work. The performance result is quite confusing for different estimators. Multinomial Naive Bayes shows the poor performance whereas the passive-aggressive shows better, but it shows some deviations. Stochastic Gradient Descent and Perceptron shows average performance. After updating datasets in the database, we are going to fit the estimators again against new datasets with a repeat of the processes. Table II shows different evaluation metrics for different estimators.

TABLE II: Evaluation Metrics of the Estimators

Estimators	**Acc.**	**Pr.**	**Rc.**	**F1**
Logistic Regression	0.88	0.85	0.91	0.88
Perceptron	0.86	0.85	0.87	0.87
Passive Aggressive	0.89	0.88	0.89	0.88
Multinomial Naive Bayes	0.87	0.87	0.87	0.87
Ensemble Using Mean	0.88	0.86	0.90	0.88

Acc. = Accuracy, Pr. = Precision, Rc. = Recall

Venkataraman et al. in article [11] used deep learning framework for English language text classification. However, they have not given any analysis report using a machine learning approach. The authors of [12], [15] used only Naive Bayes technique which was also on the English language. Authors of [15] have not used any machine learning or deep learning technique. Compared to other articles we have tried to build language independent model and gave analysis on Bengali Language data. If our framework is sequentially followed, we can classify sentiments of any language.

V. CONCLUSION

Our research gives a comparison between classification techniques for sentiment analysis. In the case of sentiment analysis, the same word can have a different sentiment. We suppose a word lust. If the sentence is ”Lust for her” then it is considered as a negative sentence but if the sentence is ”Lust for knowledge” then the sentence can be considered as a positive opinion. Hence it is quite hard building up predictor with a high level of accuracy over a large scale data containing diverse classes. Besides, nowadays text is stored in many forms like the unstructured and semi-structured format that makes it complicated to maintain this type of data and finding a pattern for them. With the fast development of data, the requirement of high accuracy of content classification is expanding. The most effective method of sentiment analysis is complicated to determine as a result shows a very diverse performance. Besides we have used translated Bengali dataset from IMDB review dataset for preprocessing which contains much noise like after translating a sentence in Bengali, there remained some English words. The work can be extended to emotion detection from text. Since sentiments are part of the basic emotions of a text.

REFERENCES

[1] J. Perkins. *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing, Birmingham B3 2PB, UK, 2014.

[2] A. A. Hakim, A. Erwin, K. I. Eng, M. Galinium, and W. Muliady. Automated document classification for news article in bahasa indonesia based on term frequency inverse document frequency (tf-idf) approach. In *2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–4, Oct 2014.

[3] KM Azharul Hasan, Sajidul Islam, GM Mashrur-E-Elahi, and Mohammad Navid Izhar. Sentiment recognition from bangla text. In *Technical Challenges and Design Issues in Bangla Language Processing*, pages 315–327. IGI Global, 2013.

[4] Princeton University. ”about wordnet.” wordnet. princeton university. 2010., October 2018. [Online]. http://wordnet.princeton.edu, [Accessed: 2019-01-10].

[5] scikit learn.org. Strategies to scale computationally: bigger data, October 2018. [Online]. http://scikit-learn.org/stable/modules/scaling_strategies.html#strategies-to-scale-computationally-bigger-data, [Accessed: 2019-01-10].

[6] Shaika Chowdhury and Wasifa Chowdhury. Performing sentiment analysis in bangla microblog posts. In *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 1–6. IEEE, 2014.

[7] Vivek Kumar Singh. Sentiment analysis research on bengali language texts. 2015.

[8] Muhammad Mahmudun Nabi, Md Tanzir Altaf, and Sabir Ismail. Detecting sentiment from bangla text using machine learning technique and feature analysis. *International Journal of Computer Applications*, 153(11), 2016.

[9] KM Azharul Hasan, Mosiur Rahman, et al. Sentiment detection from bangla text using contextual valency analysis. In *Computer and Information Technology (ICCIT), 2014 17th International Conference on*, pages 292–295. IEEE, 2014.

[10] SM Abu Taher, Kazi Afsana Akhter, and KM Azharul Hasan. N-gram based sentiment mining for bangla text using support vector machine. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE, 2018.

[11] A. Venkataraman. Deep learning algorithms based text classifier. In *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pages 220–224, July 2016.

[12] L. Jin, W. Gong, W. Fu, and H. Wu. A text classifier of english movie reviews based on information gain. In *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, pages 454–457, July 2015.

[13] S. Mandal and S. Gupta. A lexicon-based text classification model to analyse and predict sentiments from online reviews. In *2016 International Conference on Computer, Electrical Communication Engineering (ICCECE)*, pages 1–7, Dec 2016.

[14] M. Kaur and M. Bansal. Text classification using clustering techniques and p.c.a. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 642–646, Dec 2016.

[15] A. Z. Adamov and E. Adali. Opinion mining and sentiment analysis for contextual online-advertisement. In *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–3, Oct 2016.

[16] A. Rahman and U. Qamar. A bayesian classifiers based combination model for automatic text classification. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 63–67, Aug 2016.

[17] Mir Shahriar Sabuj, Zakia Afrin, and KM Azharul Hasan. Opinion mining using support vector machine with web based diverse data. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 673–678. Springer, 2017.

[18] KM Hasan, Amit Mondal, Amit Saha, et al. Recognizing bangla grammar using predictive parser. *arXiv preprint arXiv:1201.2010*, 2012.

[19] S. Raschka. *Python Machine Learning*. Packt Publishing, Birmingham B3 2PB, UK, 2015.

[20] B. Y. Pratama and R. Sarno. Personality classification based on twitter text using naive bayes, knn and svm. In *2015 International Conference on Data and Software Engineering (ICoDSE)*, pages 170–174, Nov 2015.

[21] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006.

[22] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[23] E. Diemert. Out-of-core classification of text documents, October 2018. [Online]. http://scikit-learn.org/stable/auto_examples/applications/plot_out_of_core_classification.html, [Accessed: 2019-01-10].

[24] M. Lichman. UCI machine learning repository, university of california, irvine, school of information and computer sciences, 2013. [Online]. http://archive.ics.uci.edu/ml, [Accessed: 2019-01-10].