Sentiment Analysis of Product Reviews Using Bag of Words and Bag of Concepts

Abdulwahab Almestekawy[1] and Mustafa Abdulsalam[2]
(Corresponding author: Abdulwahab Almestekawy)

Faculty of Computers and Information, Benha University
Qism Banha, Banha, Al Qalyubia Governorate, Egypt
(Email: abdulwahab.almestekawy@hotmail.com; mustafa.abdo@ymail.com)

(Received Mar. 30, 2019; Revised and Accepted July 28, 2019; First Online Aug. 21, 2019)

Abstract

Sentiment analysis is very useful for getting the overall attitude of people towards a specific topic. It can be thought of as an ability for the machine to have a common sense to judge people's opinions. This analysis becomes even more useful and efficient when the machine has access to large quantities of opinions and reviews towards the subject with which we are concerned. In this paper we focus on two quick methods of sentiment analysis and utilize them in the domain of product recommendation. The first method uses a naive Bayes classifier with bag of words approach. The second method uses a list of predefined keywords which we refer to as bag of concepts. Our results show that the proposed methods are time-efficient and can be used in online web services, while maintaining acceptable accuracy.

Keywords: Bag of Concepts; Bag of Words; Sentic Lexicon; Sentiment Analysis

1 Introduction

Sentiment analysis is a field of natural language processing (NLP) which is part of ML. It can be simply defined as the measurement of positive and negative language. Sentiment analysis helps us to automatically determine the degree to which a piece of text is positive or negative. This degree is referred to as the polarity of the text. In general, any piece of text can be either subjective (containing personal point of view), or objective (containing neutral facts). Sentiment analysis is interested in identifying subjective opinions and their polarity. The main objective of this is to determine the overall attitude and the satisfaction of people toward a specific topic or person. This can be used in many fields such as for online markets, software development companies, music brands, or even in politics. For example, sentiment analysis can be applied to customer reviews about products to help companies keep track of which products make the customers satisfied and which ones should be changed in the future. Sentiment analysis can also be used to give recommendations for new customers based on reviews of other customers who had used the products or services of a specific company.

There are challenges that face sentiment analysis. One challenge is that people tend to express multiple opinions in the same review with varied polarities. That means a single review may not be entirely positive or entirely negative. Another challenge is that some words look the same but have different meanings. So, sentiment analysis must be able to distinguish between these meanings based on other criteria, such as part of speech. Also, some expressions may be too ambiguous or too complicated for sentiment analysis to understand the meaning. The usage of metaphors, indirect speech, slang words, and misspelled words can make the task even more difficult. Sarcasm expressions also represent a challenge for sentiment analysis. In addition to all previous challenges, there are other challenges specific to each language because each language has its own syntax and ambiguities. Although there are many challenges that can prevent sentiment analysis from achieving high accuracy, sentiment analysis is still very important for many fields.

Product recommendation is an important case study in the era of e-commerce and online shopping. As more online stores are available, a customer needs to be sure that he makes the right choice when deciding between different vendors and different families of a certain product. Customers usually look for products that have been positively reviewed by other customers. Also, companies care about analyzing customer feedbacks to enhance future products and services. In order to achieve this goal, sentiment analysis has been applied to product reviews. With sentiment analysis, we can automatically detect how many positive and negative votes are there for each product without explicitly asking the users to rate the product. We can then recommend the products that have the most positive votes. We can also enable the companies to easily determine which products satisfy the customers and which ones need to be changed.

In this paper, we have utilized sentiment analysis for developing a product recommendation system based on analyzing product reviews. The rest of this paper is organized as follows: Section 2 briefly shows the related work. Section 4 describes in detail the 2 proposed methods. The results and discussion are shown in Section 5. After that we conclude the paper in Section 6.

2 Related Work

There has been many contributions by others in the field of sentiment analysis and text classification. Here we list some of these contributions and the methods used to achieve them.

In [10], The methods and results about a competition between several teams in sentiment analysis in Twitter are indicated. The competition included five sub-tasks: (i) determining the overall polarity of a tweet (3 classes: positive, negative and neutral). (ii) Determining the polarity of the tweet towards a specific topic (3 classes). (iii) Same as (ii) but with 5 classes (Including strongly negative and strongly positive). (iv) Determining the distribution of a set of tweets about a given topic across positive and negative classes. (v) Same as (iv) but with 5 classes. The first one is the most relevant to this paper. The two best ranking teams for the first sub-task in English have used deep learning approaches. One team used an ensemble of LSTMs and CNNs with multiple convolution operations. The other team used deep LSTM networks with an attention mechanism. For Arabic, the best ranking team has used naive Bayes classifier with a combination of lexical and sentiment features.

In [3], multimodal sentiment analysis has been performed using text, speech and visual cues. Bag-of-words has been used to construct the features. Classification has been carried out using a linear SVM. For text, the bag-of-words model was created using unigram and back-off bigram sequences.

In [2], SenticNet is a network of concepts arranged in 3 levels: Entity level, concept level, and primitive level. These concepts can be used in sentiment analysis. SenticNet allows sentiment analysis to be based on meanings associated with commonsense concepts instead of blindly using keywords and word co-occurrence counts. SenticNet mainly uses LSTM network to discover verb-noun primitives by lexical substitution.

In [1], it's illustrated how to use SenticNet in sentiment analysis. Sentic patterns are used to determine the overall sentiment of a text by exploiting the dependencies between words. Stanford dependency parser has been used to generate the dependency tree which is searched for linguistic patterns to extract the concepts. SenticNet lexicon is searched to get the polarity of individual concepts. After getting the polarity of individual concepts, linguistic patterns are applied to combine the polarity values into a single value representing the overall polarity. If this approach fails, an extreme learning machine (ELM) classifier is used to classify the text as positive or negative.

Bing Liu et al. have have taken part in many publications on "Opinion Spam Detection: Detecting Fake Reviews and Reviewers" such as [6]. One of the contributions of Minqing Hu and Bing Liu is the construction of an opinion lexicon that consists of two files containing about 6,800 words (concepts), one file is for negative words and the other file is for positive words. The lexicon was compiled over many years starting from their first paper [5]. In our project, we used this opinion lexicon for the second method to implement a simplified unsupervised sentiment analyzer.

In this paper, we propose two methods for product reviews sentiment analysis. The first method makes use of bag of words to train a naive Bayes classifier. The second method uses bag of concepts from Hu and Liu [7].

3 Preliminaries

3.1 Naive Bayes Classifier

The naive Bayes classifier is based on Bayes theorem. Bayes theorem uses conditional probability to determine the probability that some hypothesis (H) is true given some evidence (E). Equation (1) shows how to calculate this probability such that $P(E|H)$ is the probability that the evidence is true given that the hypothesis is true, $P(H)$ is the probability that the hypothesis is true regardless of the evidence, and $P(E)$ is the probability that the evidence is true regardless of the hypothesis.

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \tag{1}$$

In case of multiple evidence, we use the same equation but we repeat the terms $P(E|H)$ and $P(E)$ for each evidence. The probability with multiple evidence is shown in Equation (2) where n is the number of evidence.

$$P(H|e_1, e_2, \cdots, e_n) = \frac{P(e_1|H) \times P(e_2|H) \times \cdots \times P(e_n|H) \times P(H)}{P(e_1) \times P(e_2) \times \cdots \times P(e_n)} \tag{2}$$

In naive Bayes classification, we want to classify an example to one of a set of classes. Each class is considered a hypothesis, and each feature we use for classification is considered an evidence. Given a training dataset, we calculate the probability of each class, the probability of each feature, and the probability of each feature given the class for each class. Using these values, we can calculate the probability of each class given a set of features. The highest conditional probability of all classes indicates the correct class for the given example.

In sentiment analysis, we have two classes: positive and negative. The features we use are considered to be the existence of the most frequent subjective words in each sentence we classify.

3.2 Bag of Concepts

The bag of concepts method is the use of a set of common words along with their polarity values. When we want to categorize a sentence to positive or negative, we lookup the bag of concepts to get the polarity value of each word in the sentence, and we calculate the sum of these polarity values to get the overall polarity of the sentence. Equation (3) shows how to calculate the overall polarity of a sentence ($sent$).

$$\text{polarity}(sent) = \sum_{w \in sent} \text{polarity}(w) \tag{3}$$

The sign of the overall polarity determine the category of the sentence (positive, negative, or neutral).

4 Proposed Methods

4.1 Using Bag of Words and Naive Bayes Classifier

The first method is based on supervised training of a naive Bayes classifier. Our task is to classify product reviews. We used reviews from Amazon about electronics as a training dataset [9]. We have chosen the dataset such that it has 50% positive reviews and 50% negative ones. The reason for this is to prevent the classifier from being biased to a certain category of reviews. However, this has put a limit on the size of the dataset. We used adjectives and adverbs to construct the features. The first method has three main steps to prepare the classifier:

1) Cleaning and normalizing the original dataset;

2) Using the normalized dataset to construct the features;

3) Using the features to train the classifier.

The detailed steps are shown in Algorithm 1.

Before using the algorithm, we organized the dataset into a set of pairs $(review,\ class)$ Such that $class \in \{positive,\ negative\}$. The original dataset included data that we discarded such as reviewer name, reviewer id, review time, ... etc. Also the class of the review wasn't explicitly included. Instead, the dataset included an overall rating from 1 to 5 stars. We assumed that the review is positive if the number of stars is greater than 3, and is negative if the number of stars is less than 3. For reviews that have 3 stars, some reviews are positive and some are negative, so we discarded them. After organizing the dataset, we can clean and normalize the reviews. The goal of the method of cleaning and normalization is to keep only the effective words and to put these words in a unified format to eliminate the differences between the many forms of the same word, i.e. the differences between singular and plural nouns and between different conjugations of the same verb ... etc. In order to achieve this, first we tokenize the review into a list of words and we convert all words to lowercase. Then we remove the stop words (words that don't affect the polarity) from the review and select only the subjective words (words that indicate opinions). Notice that the two previous steps can be combined in a single step. In our project, the subjective words were considered to be the adjectives and the adverbs in the review. Finally we stem the words in order to normalize them before adding them to the $clean\_dataset$. Stemming was chosen instead of lemmatization because it's faster[1].

As we normalize the dataset, we make the bag_of_words by collecting all the remaining words in each normalized review to make one huge list including word duplicates. These duplicates are important to determine the number of occurrences of each word. The features are constructed based on the most frequent subjective words (most frequent adjectives and adverbs) in the normalized dataset. We constructed a frequency distribution in order to count the number of occurrences for each word and we selected the subjective words having the largest number of occurrences. Each of the words we selected represents a feature. M is the number of features we used. We decided to use 200 features. For each review, the value of each feature is either $True$ or $False$ depending on the existence of the feature word

[1]Stemming is removing prefixes and suffixes from words in order to normalize them, whereas lemmatization is converting each word to its morphological lemma

Algorithm 1 Using bag of words and naive Bayes classifier

```
1: Input: Dataset as pairs of ($review$, $class$)
2: Output: Trained naive Bayes classifier $clf$
3: const $M = 200$
4: $clean\_dataset \leftarrow [\ ]$
5: $bag\_of\_words \leftarrow [\ ]$
6: foreach ($review, class$) in dataset do
7:     $wordlist \leftarrow$ word_tokenize($review$)
8:     $wordlist \leftarrow$ to_lower($wordlist$)
9:     $wordlist \leftarrow$ remove_stopwords($wordlist$)
10:    $wordlist \leftarrow$ select_subjective_words($wordlist$)
11:    $wordlist \leftarrow$ stem_words($wordlist$)
12:    $clean\_dataset$.append (($wordlist$, $class$))
13:    $bag\_of\_words$.extend($wordlist$)
14: end
15: $FD \leftarrow$ frequency_distribution($bag\_of\_words$)
16: $feature\_words \leftarrow FD$ .get_most_common($M$)
17: $final\_dataset \leftarrow []$
18: foreach ($wordlist, class$) in $clean\_dataset$ do
19:    $features \leftarrow \{\}$
20:    foreach $feature$ in $feature\_words$ do
21:        $features$.add_key($feature$)
22:        if $feature$ in $wordlist$ then
23:            $features[feature] \leftarrow True$
24:        else
25:            $features[feature] \leftarrow False$
26:        endif
27:    end
28:    $final\_dataset$.append (($features$, $class$))
29: end
30: $clf \leftarrow$ NaiveBayesClassifier()
31: $clf$.train($final\_dataset$)
32: cache($clf$)
33: cache($bag\_of\_words$)
34: return $clf$
```

in the review. For example, if the word "good" is a feature, a review containing the word "good" will have the feature "*good*" = $True$, whereas a review that doesn't have the word "good" will have the feature "*good*" = $False$. After getting the $clean\_dataset$ and the bag_of_words, the second loop gets the feature values for each comment and append the pairs of ($features$, $class$) to the $final\_dataset$. Finally, we use the $final\_dataset$ to train a naive Bayes classifier. We didn't implement the classifier ourselves. Instead, we used NLTK naive Bayes classifier. After the training is complete, both the trained classifier object and the feature words (the names of the features) must be cached (stored in a file) to use them to classify new reviews.

When we want to classify a new review, we load the classifier and the feature words. Then we apply the same steps in Algorithm 1 to the new review in order to clean and normalize it. After that we construct the features for the new review by searching for the feature words in the normalized review. Finally we give these features to the classifier to determine whether the review is positive or negative.

4.2 Using Bag of Concepts from Hu and Liu

Instead of using bag of words, we used bag of concepts from Hu and Liu's lexicon [7]. This lexicon consists of two files. One file contains positive words, and the other file contains negative words. We assumed that each positive word has a polarity of 1, and each negative word has a polarity of -1. The general idea is to search for these concepts in the review and to calculate the sum of their polarities to determine the overall polarity of the review. At first we used Stanford CoreNLP [8] to generate a dependency tree for each review so that we can traverse this tree to extract concepts. However, the process of generating a dependency tree using Stanford CoreNLP was very slow (took an average of 90 seconds per review) and therefore wasn't applicable for online web services. For this reason, we used a more simple approach to extract the concepts.

We extract the concepts from the review by reading the words one by one from left to right and searching for each individual word in the lexicon. This linear search is very simple and it doesn't require the construction of a dependency tree. Therefore, it's very fast. However, each concept will consist of only one word (no compound words). Also, it's very difficult to detect linguistic patterns, but we used a simple method to detect negation. The negation detection is done by searching for negation words such as (not, n't, never, ... etc) and inverting the polarity of the concept that occurs after the negation word. In some cases, the concept is not directly after the negation word. For example, in the following sentence, "I'm not really satisfied with this product", there is a word between "not" and the concept "satisfied". To solve this problem, we have set a tolerance value ϵ which indicates the maximum distance between the negation word and the negated concept. If no concept is found within this distance, the negation is ignored in order to prevent the negation from affecting the wrong concept (a concept that's not related to the negation). In our project, we set ϵ to 3. Some sentences can be tricky such as "I've never seen such fantastic product!". The previous sentence is positive, but if we use the negation method we mentioned, we will apply the negation "never" to the concept "fantastic" and this will invert the polarity to negative. To solve this, we modified the negation method to ignore the negation if we find the word "such" before we find a concept.

We continue the linear search and check for negation as we extract the concepts. When the linear search is complete, the overall polarity of the review is the summation of the polarities of the concepts we found. We decide that the review is positive, negative or neutral depending on the sign of the overall polarity (+ve, -ve, or 0 respectively). The steps of this method are summarized in Algorithm 2.

Algorithm 2 Using bag of concepts from Hu and Liu

```
Input: $review$ to calculate its polarity
Output: $polarity$ of the $review$
const $\epsilon = 3$
$words \leftarrow$ word_tokenize($review$)
$polarity \leftarrow 0$
$negation\_flag \leftarrow False$
$negation\_distance \leftarrow 0$
foreach $word$ in $words$ do
    if is_negation_word($word$) then
        $negation\_flag \leftarrow True$
        $negation\_distance \leftarrow 0$
    else
        if $word$ is "such" then
            $negation\_flag \leftarrow False$
            $negation\_distance \leftarrow 0$
        else
            if $negation\_flag$ is $True$ then
                $negation\_distance \leftarrow negation\_distance + 1$
            end
            if $negation\_distance > \epsilon$ then
                $negation\_flag \leftarrow False$
                $negation\_distance \leftarrow 0$
            end
            $p \leftarrow 0$
            if $word$ in $positive\_words\_file$ then
                $p \leftarrow 1$
            end
            if $word$ in $negative\_words\_file$ then
                $p \leftarrow -1$
            end
            if $negation\_flag$ is $True$ and $p \neq 0$
                $p \leftarrow p \times -1$
                $negation\_flag \leftarrow False$
                $negation\_distance \leftarrow 0$
            end
            $polarity \leftarrow polarity + p$
        end
    end
end
return $polarity$
```

5 Results and Discussion

5.1 Data Description

The datasets we used were collected from data.world [4] and amazon [11], [9]. At first we collected 300,000 reviews:

1) 70,000 reviews about various products from [4];

2) 30,000 reviews about an mp3 player from [11];

3) 100,000 reviews about electronics and 100,000 reviews about cell phones from [9].

The datasets didn't include explicit polarity labels for the reviews. However, they included a 5-star rating. We used this rating to generate the polarity labels before using the datasets. We assumed that a review is positive if it has more than 3 stars, and is negative if it has less than 3 stars. The remaining reviews that had 3 stars are mixed (they include both positive and negative reviews), so we removed these reviews from the datasets and used the remaining reviews. For this reason, the size of the datasets was reduced to the following:

1) 65,677 reviews about various products from [4];

2) 27,566 reviews about an mp3 player from [11];

3) 92,018 reviews about electronics and 88,264 reviews about cell phones from [9].

One problem in the previous datasets was that they had very large number of positive reviews compared to negative reviews. This resulted in biased (fake) accuracy. In order to get the real accuracy we had to make the number of positive reviews equal to the number of negative ones. For this reason, we reduced the datasets again to meet this condition. The size of the non-biased datasets is as follows:

1) 11,068 reviews about various products from [4];

2) 12,292 reviews about an mp3 player from [11];

3) 22,652 reviews about electronics and 28,330 reviews about cell phones from [9].

Table 1 shows the formats of the datasets we used. The datasets from [4], and [9] were in csv and json formats respectively. The dataset from [11] was not in a standard format but was converted to csv.

Table 1: Dataset formats

Dataset	Format
products [4]	csv
mp3 player [11]	csv
amazon [9]	json

5.2 Parameter Setting

The values for the parameters we used for our algorithms are shown in Table 2 where M is the number of features for the first algorithm, and ϵ is the maximum number of words between the negation word and the concept affected by the negation.

Table 2: Parameters of the algorithms

Algorithm	Parameter	Value
Bag of words (BoW)	M	200
Bag of concepts (BoC)	ϵ	3

5.3 Results

Before discovering the problem of the biased datasets, we used the biased datasets for training and testing. The naive Bayes classifier for the first method was trained using 92,018 reviews about electronics from [9]. The second method requires no training. Both methods were tested using the rest of the biased datasets. The accuracy of the first method, bag of words (BoW), was calculated as the ratio between the number of correctly classified reviews to the total number of reviews. However, the second method, bag of concepts (BoC), has 3 outcomes: positive, negative, or neutral. The reviews categorized as neutral are treated as if they weren't categorized, i.e. they won't change the number of positive or negative votes and hence should be ignored. In this case, we subtract the number of reviews categorized as neutral from the total number of reviews before calculating the accuracy of the BoC method. Equations (4), (5) illustrate how we've calculated the accuracy of BoW and BoC methods respectively.

$$BoW\_accuracy = \frac{correct}{total} \tag{4}$$

$$BoC\_accuracy = \frac{correct}{total - neutral} \tag{5}$$

It was found, after ignoring neutral reviews, that the BoC method has higher accuracy than BoW method. However, the execution of BoW method is slightly faster. The results of both methods using the biased dataset are shown in Table 3. Keep in mind that the accuracy values yielded by the biased datasets are not the actual accuracy values due to the imbalance between the number of positive and negative reviews. Figures 1, 2 visualize the difference between the two methods in accuracy and execution time respectively using the biased datasets.

Table 3: Results using biased datasets

dataset	size	BoW accuracy	BoC accuracy	BoW avg time (ms)	BoC avg time (ms)
products	65,677	91.74%	91.72%	7	6
mp3 player	27,566	80.09%	86.88%	19	26
cell phones	88,264	84.10%	88.58%	11	14
Overall	181,507	86.26%	**89.42%**	**11**	13

To have more realistic insight of the accuracy, we had decided to use non-biased datasets in which the number of positive reviews equals the number of negative ones. This ensures if the classifier, for example, categorizes all reviews as positive, then the accuracy won't exceed 50% to indicate that there's something wrong with the classifier, unlike the biased datasets where the same situation would yield a high accuracy. The size of the non-biased training dataset for the BoW method was 22,652. The rest of the non-biased datasets were used to test both methods. Table 4 shows the results after using the non-biased datasets. The results show that the accuracy was reduced but became more realistic, and this is an indicator that the biased dataset was not giving the actual accuracy. The execution time however didn't change a lot. The results still indicate that the BoC method has higher accuracy

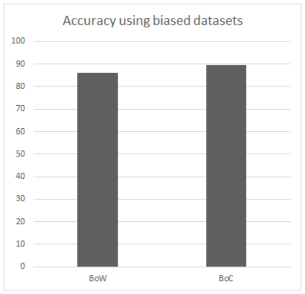

Figure 1: Accuracy of the two proposed methods using the biased dataset

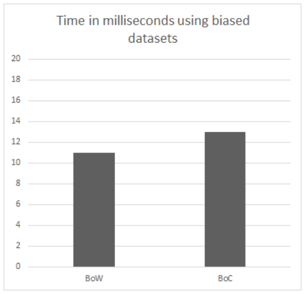

Figure 2: Average execution time for each of the two proposed methods using the biased dataset

than the BoW method, whereas the BoW method is slightly faster than the BoC method. Figures 3, 4 visualize the difference between the two methods in accuracy and execution time respectively using the non-biased datasets.

Table 4: Results using non-biased datasets

dataset	size	BoW accuracy	BoC accuracy	BoW avg time (ms)	BoC avg time (ms)
products	11,068	70.57%	72.48%	8	8
mp3 player	12,292	75.38%	74.88%	20	27
cell phones	28,330	67.85%	75.30%	12	15
Overall	51,690	70.22%	**74.62%**	**13**	16

6 Conclusion

Sentiment analysis of product reviews can be very useful for both customers and companies. We have implemented 2 methods for sentiment analysis: (i) Using bag of words and naive Bayes classifier. (ii) Using bag of concepts from Hu and Liu. We also showed the difference between bag of words and bag of concepts, and the difference between supervised and unsupervised classification. The results show that although the second method was very straightforward, it achieved higher accuracy than the first method. However, the methods we introduced can still be tricked by ambiguous sentences and complex linguistic patterns. Some techniques such as sarcasm detection and predicting missing parts

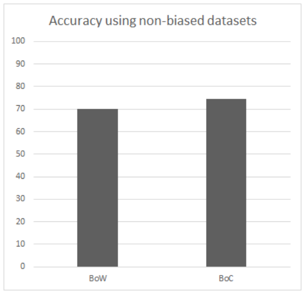

Figure 3: Accuracy of the two proposed methods using the non-biased dataset

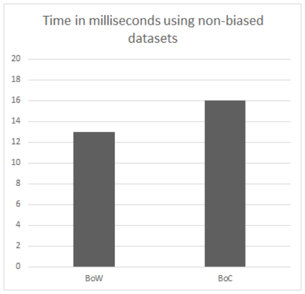

Figure 4: Average execution time for each of the two proposed methods using the non-biased dataset

of sentences were not considered for the sake of simplicity and time efficiency. Hopefully, the majority of product reviews are subjective, not ambiguous, and not sarcastic. So, a high accuracy can still be achieved. But if we wish to achieve even higher accuracy, more advanced approaches should be used, but it's important to balance between accuracy and performance while implementing new methods.

References

[1] E. Cambria and A. Hussain, "Sentic Patterns," in *Sentic Computing A Common-Sense-Based Framework for Concept-Level Sentiment Analysis*, vol. 1, Springer, pp. 73–106, 2015.

[2] E. Cambria, S. Poria, D. Hazarika, and K. Kwok, "SenticNet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

[3] N. Cummins, S. Amiriparian, S. Ottl, M. Gerczuk, M. Schmitt, and B. Schuller, "Multimodal Bag-of-Words for Cross Domains Sentiment Analysis," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'18)*, pp. 1–5, 2018.

[4] Datafiniti, "Grammar and Online Product Reviews - dataset by datafiniti," *Data World*, Apr-2018. (`https://data.world/datafiniti/grammar-and-online-product-reviews`)

[5] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 168–177, 2004.

[6] H. Li, G. Fei, S. Wang, B. Liu, W. Shao, A. Mukherjee and J. Shao, "Bimodal Distribution and Co-Bursting in Review Spam Detection," in *International World Wide Web Conference*, Perth, Australia, 2017.

[7] B. Liu and M. Hu, "Opinion Mining, Sentiment Analysis, Opinion Extraction," *UIC Computer Science*, 15-May-2004. (`https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`)

[8] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55-60, 2014.

[9] J. McAuley, "Amazon product data," *Amazon review data*, 2014. (`http://jmcauley.ucsd.edu/data/amazon/`)

[10] S. Rosenthal, N. Farra, and P. Nakov, "SemEval-2017 Task 4: Sentiment Analysis in Twitter," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 502–518, 2017.

[11] TIMAN, "Review data sets for 'Latent Aspect Rating Analysis," *TIMAN Group*, Mar. 2018. (`http://times.cs.uiuc.edu/~wang296/Data/`)

Biography

Abdulwahab Almestekawy was born on October 13, 1996 in Benha, Egypt. He has been interested in programming and game development since he was 13. He has graduated from high school in 2015 with grade 99.26%. He has joined the faculty of computers and information of Benha university and has currently finished his third year of study. He became interested in artificial intelligence (AI) and machine learning (ML) in college, and has made a project on sentiment analysis for online product reviews. (Mobile: +201091749989).

Mustafa Abdul Salam was born on November 1, 1981 in Sharkia, Egypt. He received the B.S from Faculty of Computers and Informatics, Zagazig University, Egypt in 2003, and obtained master degree in information system from faculty of computers and information, Menufia university, Egypt in 2009 specializing in Hybrid Machine Learning and Bio Inspired Optimization algorithms. He obtained his Ph.D. degree in information system from faculty of computers and information, Cairo University, Egypt. He is currently a Lecturer in faculty of Computers and Information, Benha University, Egypt. He has worked on a number of research topics .Mustafa has contributed more than 20+ technical papers in the areas of neural networks , support vector machines, optimization , time series prediction , extreme learning machine , hybrid CI models in international journals, international conferences, local journals and local conferences. His majors are Machine Learning, Big Data, Stream Data Mining, and Deep Learning.