Sentiment Analysis with Naïve Bayes over Internet Movie Database (IMDb) Movie Reviews

Akbar Putra Novial
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia
akbar.putra@ui.ac.id

Avatar Putra Pertama Azka
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia
avatar.putra@ui.ac.id

Johanes Steven
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia
johanes.steven@ui.ac.id

Kevin Constantine
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia
kevin.constantine@ui.ac.id

Mohammad Saddam Mashuri
Faculty of Computer Science
Universitas Indonesia
Jakarta, Indonesia
mohammad.saddam91@ui.ac.id

***Abstract*—The public opinion is an essential factor to take note of, especially for businesses that involve an exchange of goods and services for monetary value. Sentiment analysis is a technique to analyze public sentiments for business owners to make decisions that may heavily influence their personal and company livelihoods. Social media and review websites play a huge role in delivering the opinions of the public square to the business owners and, as such, can be analyzed. This paper aims to analyze IMDb movie reviews. As one of the most popular online databases, the movies receive many reviews from people. By doing sentiment analysis on the reviews, we can differentiate positive reviews from negative ones. This helps in judging whether the movie's overall evaluation is considered good or bad by the viewers. One such technique to analyze public sentiments is the Naïve Bayes, a probabilistic algorithm based on Bayes' theorem that can be trained using the IMDb sentiment dataset and used to predict a movie review's sentiment.**

***Keywords*—Public opinion, sentiment analysis, IMDb movie reviews, Naïve Bayes**

I. INTRODUCTION

In this day and age, technology has become an essential part of our lives. Businesses all over the globe have also utilized technology to retrieve various amounts of data that can support them. With a large amount of data that flows each day, there has to be a way to help them manage and monitor the data. Sentiment analysis is one of them. Sentiment analysis is a technique used by business owners to help in monitoring public sentiments and understanding what their customer needs so that they may make accurate judgments to help their businesses grow.

One such technique of sentiment analysis is the Naïve Bayes, which involves having a probabilistic algorithm being trained on a categorized dataset (positive or negative) to make predictions on the given text's category. This technique relies on the Bayes' Theorem, a statistical technique named after its inventor, Reverend Thomas Bayes, that is able to calculate the probability of an event to occur, given a certain probabilistic condition to have occurred [1].

In this modern era, the sentiments of the public can be delivered straight to the business owners with the help of the web. One such example of this is the IMDb movie reviews.

IMDb is an online movie database storing titles of movies, television shows, video games, and other such entertainment sources, including those who contributed in their making, such as the director, producer, and actors. [2].

IMDb provides a place where people who have consumed those entertainment sources publish their opinions and ratings of those entertainment sources to be displayed to the whole world. Other people who have not encountered those entertainment sources can then base their judgments on whether or not to bother with them on these reviews. Thus, these reviews can become an important source of data for movie publishers to understand whether or not their products are well-liked.

This study aims to train a Naïve Bayes algorithm with a dataset of movie reviews that have been categorized into two groups: positive reviews and negative reviews. The algorithm will judge whether or not a given input is positive or negative. This information can then be helpful for movie directors to get a quick insight into how well-received their movies are by the general public, which could then be used to make business decisions based on those gathered insights.

We specifically chose the Naïve Bayes approach to this problem, due to its efficiency and effectiveness in determining text-based sentiments. By virtue of its algorithm—that is, taking independent features from the dataset and chaining Bayes' Theorem to determine the probability of the overall element fitting into a given class—Naïve Bayes offers a much simpler way for us to classify these user sentiments when compared to other methods such as logistic regression. This is attributed to Naïve Bayes assuming these features to be independent of each other.

Following the Introduction section, Section II will provide the preliminaries, Section III will describe the methodology of the experiment, Section IV will provide the results of the experiment and the analysis, while Section V will provide the conclusions of this experiment.

II. RELATED WORK

Pang et al. [3] introduced the concept of classifying documents not by the topic of the document itself but by the overall sentiment it received, such as its received positive and negative reviews. The authors utilized standard machine learning techniques in order to analyze movie reviews and

found out that the applied machine learning techniques outperformed the baseline human ability to classify documents based on sentiments.

Pang and Lee [4], delved into the broad topic of sentiment analysis and opinion mining in a time where online reviews and personal blogs were growing more and more popular. They developed a methodology for sentiment analysis, such as calculating the ratio between the positive to the negative words.

Pak and Paroubek [5], utilized Twitter as a source of opinion mining in 2010. Twitter has millions of users utilizing its services every day and, as such, is an excellent source of corpus collection for sentiment analysis. From the collected corpus, the authors can build a classifier that can categorize positive, negative, and neutral sentiments.

Read [6] dug deeper into the sentiment analysis methodology, managing to account for emoticons, such as the 'smiley face' and the 'sad face' emoticon in the dataset. The dataset is split into two, positive and negative, based on the type of emoticons used and is then used to train SVM and Naïve Bayes classifiers.

III. METHODOLOGY

A. Problem Statement

Movie ratings are a useful metric that can be utilized by both consumers and producers of the film. A consumer may have a look at a movie's ratings before deciding to watch it. Likewise, a producer will want to know how well their work is doing at the box office.

In terms of Naïve Bayes, we can adequately define the aim of this sentiment analysis problem as having to determine the positive or negative sentiment behind a given written movie review.

A positive movie review is a review that shows that the reviewer had a good experience while watching the movie. A good review may praise parts of the movie such as the story, the production of the movie, the acting, etc. One such example is "This show was an amazing, fresh, and innovative idea."

On the other hand, a negative review shows that the viewer had a negative experience. An example of a negative review is "I had the terrible misfortune of watching this movie in its entirety."

We recognize that, in reality, some movie reviews may not fall neatly into either of these categories, though this method of classification provides a simple, comprehensive view of general audience sentiment.

B. Experiment Workflow

Approach. We developed our solution based on the Naïve Bayes algorithm using Python.

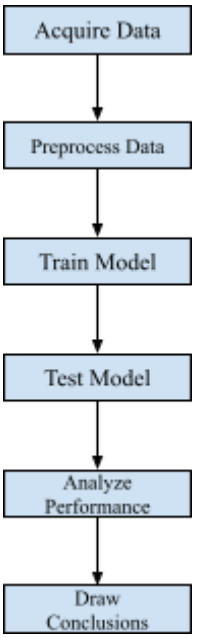

Fig. 1. Flow Diagram of the Experiment

Our experiment can be broken down into these parts:

1. Acquire Data: Acquire the dataset from a source.
2. Preprocess Data: Preprocess the data for unneeded words and characters to increase the accuracy of our model during the training phase.
3. Train Model: Create a bag of words from positive and negative reviews from the training dataset and fit it into the model.
4. Analyze Performance: Test the model to see if it can correctly identify the labeled positive and negative reviews.
5. Draw Conclusions: Analyze the result, visualize in the form of a graph, and draw conclusions from the result.

C. Dataset

The dataset this paper operates upon is taken from an open dataset provided by Stanford University [7], as detailed in its accompanying paper [8]. As this is a classification problem, labeled data is required to train the model.

The dataset consists of a set of 25,000 labeled movie reviews for training and another labeled set of 25,000 for testing. The labeled data consists of 12,500 positive and negative reviews. The testing will compare the true label with the predictive label produced by our model.

D. Experiment Setup

Preprocessing. Before creating the model, it is necessary to first process the data in order to make it machine-readable. The efforts put into preprocessing the data include the removal of artifacts such as HTML tags and transforming strings into lower case. Furthermore, any contractions in writing are expanded into their complete forms to create a more uniform word list (e.g., I've → I have, Don't → Do not, Won't → Will not, etc.).

Then, we clean the input from any non-alphabet characters, as they are treated as trivial for sentiment analysis. It is important to note that the removal of non-alphabet characters is done after the expansion of contractions to avoid breaking up contractions (i.e., “won’t” into “won t”).

Lastly, stop words are removed, and a lemmatizer is implemented, which converts the word into its base form (e.g. words “Targets”, and “Target”, are considered the same) to reduce data variants. We have opted not to consider the word “not” as a stop word as it may play a significant role in determining negative reviews.

Here is a visualization of a sentence with and without the preprocessing:

TABLE I. EXAMPLE OF PREPROCESSING

Feature	Sentence
Original	 This is one of the comparison examples which I’ve come up with!
Removing HTML Tag	This is one of the comparison examples which I’ve come up with!
Fixing Contractions	This is one of the comparison examples which I have come up with!
Removing non-alphabet	This is one of the comparison examples which I have come up with
Removing stop words	one comparison examples come with
Lemmatize and the final result	one comparison example come with

E. The Algorithm

Basis of Naïve Bayes. Following the preprocessing stage, we use the Naïve Bayes theorem below to find the probability of a particular sentence within a category, namely positive and negative reviews. The Naïve Bayes Theorem is simple, fast, has high accuracy, and works well on text classification [9]. Furthermore, it only requires a small amount of training data to estimate the parameters necessary for classification [10]. The category with the highest classification probability will be selected.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In this figure, P(A|B) represents the probability of A given the event B. We derive said probability from the initial probability of A multiplied by the probability of B given A. Then it is divided by the probability of B. Note that this base formula is modified for better performance and to account for other factors specific to text classification.

A common pitfall when performing text classification using Naïve Bayes is that of zero-frequency values. These occur when a certain word has no entry in a class, thus reducing the probability of the whole string fitting into a certain class to be zero. To avoid such a computational problem, we add an additional frequency count to all word frequencies on all categories for each time a word didn’t appear on every category, also known as additive/Laplace smoothing.

Here we utilized a variation of Naïve Bayes called Multinomial Naïve Bayes. In this variety, each probability of a word being in a sentiment class p(word|class) is assumed to follow a multinomial distribution, which is a type of probability distribution where there are more than two involved variables. This works particularly well when dealing with word counts in text classification, as word counts naturally follow a multinomial distribution. This behavior is what makes this variation suitable for text-based sentiment analysis, our problem at hand.

Implementation. We implemented this algorithm using the Python library scikit-learn. The following code snippets provide a brief description of our implementation.

```
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Preprocess reviews, omitted for brevity
# store list of preprocessed reviews as variable
“data”

# Create bag of words
vect = CountVectorizer()
count = vect.fit_transform(data)
#target is the category of each train dataset
# Fit to MultinomialNB model
clf = MultinomialNB().fit(count, target)
```

Here data represents the list of preprocessed reviews. We then create a bag of words matrix from that list. This essentially lets us keep track of the word counts for each class. With this bag of words, we can train our model to predict the sentiment of future reviews based on the Multinomial Naïve Bayes algorithm.

```
# Fit testing dataset to variable x
x = vect.transform(testing_dataset)

# Predict sentiments of x based on model
predicted = clf.predict(x)
```

We use the trained model to predict the testing dataset. The list of results is stored in the variable “predicted.” We can then compare the results with the actual classification (i.e., positive or negative). We will discuss the results of this algorithm in the next section.

The full code to our experiment program can be found in the following repository:
https://github.com/Constantine-Kevin/ai_ds_project

IV. RESULTS AND DISCUSSION

Now, we would like to show the result of our experiment. We ran the algorithm on several types of preprocessing variants, one which doesn't involve any preprocessing, one which only involves a lemmatizer, one which only involves removing stop words, and lastly, one which uses both lemmatization and removal of stop words.

In the following table, we use the measures of accuracy and precision. These following formulae determine these measures:

$$Accuracy = \frac{|True\ Positive| + |True\ Negative|}{|Training\ Dataset|}$$

$$Precision = \frac{|True\ Positive|}{|Positive\ Dataset|}$$

Where true positive indicates reviews that have been correctly identified as positive, and so forth. Please note that the term "Precision" refers to the positive precision, also known as the Positive Predictive Value (PPV). Its negative counterpart, Negative Predictive Value (NPV) follows a similar formula, adjusted for the negative dataset.

We run the algorithm along with the variants on the test dataset, which includes 12500 positive and 12500 negative reviews which our model has classified. Below is a table with the results, along with the resulting accuracy, and precision of the model.

TABLE II. EXPERIMENT RESULTS

Feature	Results	Accuracy & Precision
Without lemmatizer and removing stop words	Positive: 9341 / 12500 Negative: 10987 / 12500	Accuracy: 81.312% Precision: 74.728%
With lemmatization only	Positive: 9310 / 12500 Negative: 10967 / 12500	Accuracy: 81.108% Precision: 74.480%
With removing stopword only	Positive: 9579 / 12500 Negative: 11028 / 12500	Accuracy: 82.428% Precision: 76.632%
With both lemmatizer and removing stop word	Positive: 9534 / 12500 Negative: 11005 / 12500	Accuracy: 82.156% Precision: 76.272%

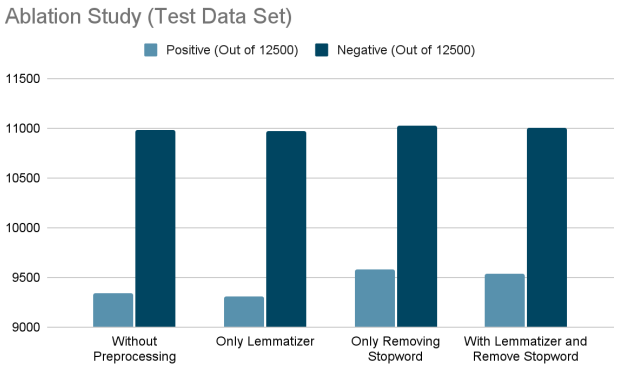

Fig. 2. Graph of Ablation Study

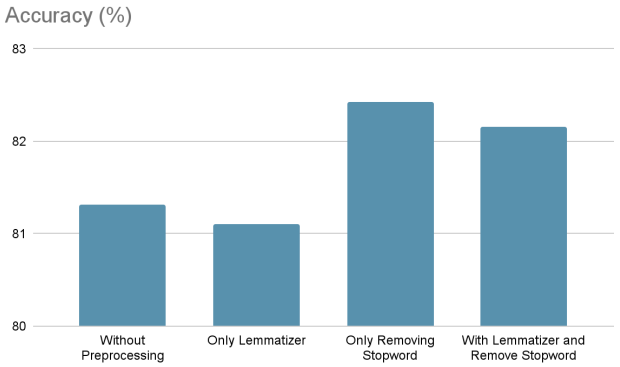

Fig. 3. Graph of Accuracy Comparison

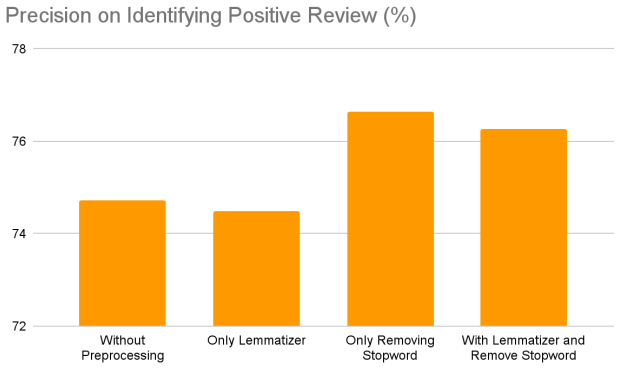

Fig. 4. Graph of Precision Comparison

From the figure above, it has been shown that the preprocessing part plays a role to improve the accuracy of the model from a base of 81.312% to a maximum of 82.428% by only removing stopwords.

From Table II, we found that the model tends to predict negative reviews more accurately than positive ones, as can be seen from the higher average NPV (87.974%) compared to the average PPV (75.528%). This means that, on average, the NPV is 16.478% higher than the PPV.

Within our ablation study, we found that removing stop words is beneficial to the accuracy of the model, as it increases the accuracy by contributing to a maximum of

1.116% increase in the accuracy. On the other hand, lemmatizer plays a role in decreasing the model's accuracy by a maximum of 0.272%. We suspect that it could be due to details that are cut off from the sentences, which might have played a role in determining the category.

V. CONCLUSIONS

From our findings, we are able to conclude that our Naïve Bayes model is able to predict both positive and negative labels of movie reviews accurately. Moreover, the model can predict negative labels more precisely than positive labels within a reasonable margin.

Furthermore, we have also discovered that the removal of stopwords is beneficial to the performance of our model. The lemmatization of reviews, on the other hand, proved to be detrimental.

For future research, we would like to run the model several times to test its consistency. In the future, we would like to find out why stop words and lemmatization might affect the accuracy of the model. We would also like to find more ways to optimize this algorithm, such as improving the precision.

REFERENCES

[1] R. Routledge, "Bayes's theorem | Definition & Example", Encyclopedia Britannica, 2021. [Online]. Available: https://www.britannica.com/topic/Bayess-theorem. [Accessed: 26- Jun- 2021].

[2] "IMDb | Help", Help.imdb.com, 2021. [Online]. Available: https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpsect_cons_1_1#. [Accessed: 26- Jun- 2021].

[3] B. Pang, L. Lee and S. Vaithyanathan, "Thumbs up?", Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02, vol. 10, pp. 79-86, 2002. Available: https://dl.acm.org/doi/10.3115/1118693.1118704. [Accessed 26 June 2021].

[4] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis", Foundations and Trends® in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008. Available: https://www.nowpublishers.com/article/Details/INR-011. [Accessed 26 June 2021].

[5] A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", in Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, Valletta, 2010.

[6] J. Read, "Using Emoticons to Reduce Dependency in Machine Learning Techniques for Sentiment Classification", Proceedings of the ACL Student Research Workshop, pp. 43-48, 2005. Available: https://www.aclweb.org/anthology/P05-2008/. [Accessed 26 June 2021].

[7] "Sentiment Analysis", Ai.stanford.edu. [Online]. Available: https://ai.stanford.edu/~amaas/data/sentiment/. [Accessed: 26- Jun- 2021].

[8] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng and C. Potts, "Learning Word Vectors for Sentiment Analysis", in The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, 2011, pp. 142-150.

[9] L. Dhande and G. Patnaik, "Analyzing Sentiment of Movie Review Data using Naive Bayes Neural Classifier", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 3, no. 4, 2014. Available: https://www.semanticscholar.org/paper/Analyzing-Sentiment-of-Movie-Review-Data-using-Dhande-Patnaik/506ce02487e90016eab07bc76090905b5e6e9365#citing-papers. [Accessed 26 June 2021].

[10] L. Dey, S. Chakraborty, A. Biswas, B. Bose and S. Tiwari, "Sentiment Analysis of Review Datasets Using Naïve Bayes' and K-NN Classifier", International Journal of Information Engineering and Electronic Business, vol. 8, no. 4, pp. 54-62, 2016. Available: https://www.researchgate.net/publication/305001704_Sentiment_Analysis_of_Review_Datasets_Using_Naive_Bayes%27_and_K-NN_Classifier. [Accessed 26 June 2021].