IT1244 Project Report: Sentence Sentiment

National University of Singapore
Brian Bong Neng Ye, Ng Yu Wen, Chin Jie Yi, Lee Jie Shi

1 Introduction

In the rapidly evolving landscape of digital communication, sentence sentiment analysis has become an important tool for understanding and interpreting the vast amounts of textual data generated daily online (Gunasekaran 2023). This process involves classifying sentences according to their sentiment, typically categorized as positive or negative. Given the surge in online content such as customer reviews, the ability to assess the sentiment of a sentence automatically has significant implications among businesses and organizations to better understand customer feedback, market research, and identify trends. By analyzing relevant data, businesses may gain valuable insights which would in turn guide the business to make informed decisions.

However, the path to achieving high accuracy in sentence sentiment analysis is full of challenges due to the inherent complexity of human language. Traditional methods, which mainly rely on manual features or simple lexical approaches, often fail to capture the nuanced expressions of sentiment. Recent advancements in AI/ML, particularly in Deep Learning and natural language processing (NLP), have opened new opportunities for tackling this problem more effectively.

Reviewing past research regarding sentiment analysis, we examined ready-made sentiment libraries developed by researchers. This includes VADER (Valence Aware Dictionary for sEntiment Reasoning), a model developed by Hutto and Gilbert (2014). It provides a sentiment score for a given text according to a dictionary of words and rules, using valence scores for every single word to determine its positivity or negativity. However, the model is generally tailored to the lexicons and language structures commonly found in social media, which is mostly short, informal sentences.

We have also reviewed TextBlob, another Python library designed to process textual data for natural language processing tasks. Specifically, for sentiment analysis, TextBlob uses pattern-based analysis where the sentiment property returns polarity or subjectivity. We examined a model where TextBlob and Deep Learning are integrated to analyze social media comments with regards to the airline industry in the US (Aljedaani et al. 2022). The model yielded a high accuracy in the predictions of the sentiments of comments. However, it warns against possible mislabelling of data as a potential drawback regarding the accuracy of the model, suggesting the need to manually review the outcomes of artificial models.

Considering limitations gathered from the two reviews, we recognize that most sentiment analysis models are trained on social media data, as it is in the case of VADER and TextBlob, thus the possible inaccuracies in adopting these libraries into our dataset. Furthermore, we also note the possible mislabelling of the dataset, thus by suitably outputting samples of data during the development of the model, we aim to minimize, or at least recognize, some inaccuracies of labeling that may occur in the dataset.

2 Dataset

The dataset used consists of text data paired with sentiment labels, where 1 indicates positive sentiment and 0 indicates negative sentiment. The dataset is balanced, with equal amounts of observations with positive sentiment and negative sentiment.

The distribution of sentence lengths for each sentiment class are plotted. As reflected in the histogram below, the distribution reveals no significant differences between the length of sentences of positive sentiment and negative sentiment.

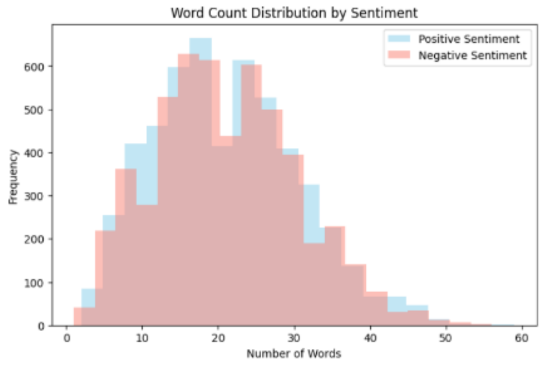


Figure 1: Word Count Distribution by Sentiment

Furthermore, word clouds for positive and negative reviews that visually represent the most common words in each sentiment category are generated. This analysis highlights the presence of particularly indicative words of sentiment, besides certain words like “film”, “movie” or “one” that do not necessarily indicate any sentiment as it is represented heavily in both cases.

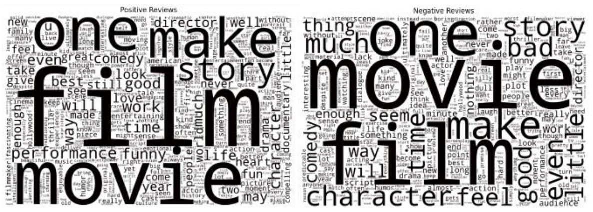


Figure 2: Word Clouds of Different Sentiments

Thus, we identify certain limitations of the dataset that may impact the effectiveness of a model, including:

- **Size:** With only 10,000 samples, it is too small for training language models from scratch, which normally requires larger amounts of data to understand the nuances of natural language adequately, e.g. BoW (Bag of Words).
- **Noise:** In this context, we define noise as possible semantic errors that may arise in the sentence, including spelling errors and foreign language. This may affect the model's ability to generalize well to new data (Sharou et al. 2021).
- **Repetitive Words:** Certain words such as "film" and "movie" are represented extensively in both sentiments. Such words may not be significantly useful for the model to identify sentiments.

Given the limitations, leveraging external data becomes essential. One strategy involves using pre-existing word embeddings, where we used two embeddings, particularly Word2Vec and TF-IDF in our project.

Word2Vec

Developed by a Google team, Word2Vec employs techniques such as CBOW (Continuous Bag of Words) and Continuous Skip-Gram Model. The embedding has a better ability in capturing semantic relationships and being flexible to linguistic complexity (Mikolov et al. 2013). We implement Word2Vec by importing the library "gensim".

TF-IDF

TF-IDF (Term Frequency Inverse Document Frequency) assigns a weight to each term in a document, reflecting both its frequency within the document and its importance in the corpus. By balancing higher frequencies of certain terms by penalizing its occurrences across sentences, it assigns more weight to terms which happen more frequently in certain sentences only (Kim and Gill 2019). TF-IDF is implemented by the library "sk-learn" in our model.

In the process of word embedding, tokenization, lemmatization and removal of stop words are done, which could potentially minimize the influence of noise on the model.

3 Methods

The resultant vectors are split into training and testing datasets according to a ratio of 8:2 after embedding the dataset. Our goal is to classify textual data into sentiment categories of 1 (positive sentiment) and 0 (negative sentiment) based on their content.

Due to the limitations of data size, we did not consider Deep Learning as one of our methods when doing sentence sentiment analysis. Deep learning models are known for requiring large amounts of labeled data to effectively learn the underlying patterns of the dataset. With only 10,000 samples, we could not get any significant insights from utilizing deep learning (Benkendorf and Hawkins 2020). With smaller datasets, simpler supervised learning models such as Naive Bayes, Logistic Regression, or even tree-based models like Random Forest can perform comparably or sometimes even better than deep learning models. These models are faster to train and less prone to overfitting with limited data, offering a better trade-off between performance and resource utilization (Rudin 2019).

Therefore, we employ the following models, namely Logistic Regression, Random Forest Classifier, Extra Trees Classifier and Naive Bayes Classifier. We will further explain decision trees and Naive Bayes Classifier which are not taught in this course. Additionally, we used the pre-trained model TextBlob and VADER to compare the result with our model.

Logistic Regression

Logistic Regression utilizes the sigmoid function $g(x) = 1/(1+e^{-x})$ to calculate the class probability of a sample. It is employed to investigate how predictor variables impact categorical outcomes, typically involving binary outcomes in high-dimensional spaces (Nick et al. 2007), thus making it suitable for our binary classification task. We used "sk.learn" to import the algorithm from the "models" module. All parameters were kept at the default values.

Decision Trees

Decision trees represent a series of decisions and their possible consequences in the abstract form of trees. A decision tree consists of nodes, branches, and leaves. Each node represents a decision made based on one or more input features, each branch represents a possible outcome of that decision, and each leaf node represents the final prediction or decision. The goal of decision trees is to maximize the information gain or minimize the impurity of the data. This is achieved by recursively splitting the dataset into smaller subsets based on the values of the input features (Loh 2011).

Decision trees can identify key features, which may be helpful in our case as certain words may be highly indicative of a certain sentiment, such as “fabulous” for positive sentiment. However, in usual decision trees, overfitting is prone to happen as the tree makes a more complicated series of decisions in the splitting process. Hence, we explored two variations of decision trees, Random Forest and Extra Trees. These two algorithms are implemented by importing from the ‘sk-learn.ensemble’ module with default parameters.

Random Forest Classifier

Random Forest is an ensemble learning technique that constructs a multitude of decision trees at training time and outputs the sentiment class. The algorithm introduces randomness into the model by building each tree on a random subset of data and considering a random subset for each split decision. The final prediction is made by combining the predictions of all trees in the forest, typically through majority voting for classification tasks. The model can handle high-dimensional data well, making it suitable when doing sentiment analysis when features for instance, words are numerous (Benkendorf and Hawkins 2020).

Extra Trees Classifier

The Extra-Trees Classifier is another ensemble learning method that builds a collection of unpruned decision trees using the traditional top-down approach. It randomly selects cut points for node splits and uses the entire learning sample, rather than bootstrap replicas, to grow the trees (Ali et al. 1996). This could improve the model’s overall predictive performance and robustness. The final prediction is determined by averaging the predictions across all trees using a majority vote for the classification problem. We employed this model as it often achieves decent predictive performance with less computational cost, since it eliminates the need for searching for optimal split points.

Naive Bayes Classifier

Naive Bayes Classifier is based on applying Bayes’ theorem, with the “naive” assumption of conditional independence between every pair of features given the class label. In the model, each input is considered independently, which simplifies the computational time of the likelihood of the data given the class label. The classifier calculates the probability of each class given the input features and selects the class with highest probability as prediction (Wickramasinghe 2021). For sentiment analysis, it calculates the probability of each class under the condition of a given feature set (word occurrences), using frequency counts. While assuming independence across all words in the data, the model is used widely in NLP due to its efficiency and good performance.

For TF-IDF embedding, we perform a variation of Naive Bayes known as Multinomial Naive Bayes, which assumes multinomial distribution of features that can be aptly described by the frequencies of words in sentences (Xu et al. 2017). For Word2Vec embedding, since Multinomial Naive Bayes does not fit continuous values, Gaussian Naive Bayes was fitted instead. To implement the model, we imported the algorithm from the module “sklearn.naive_bayes”.
The following flowchart that summarizes our methodology for the model development.

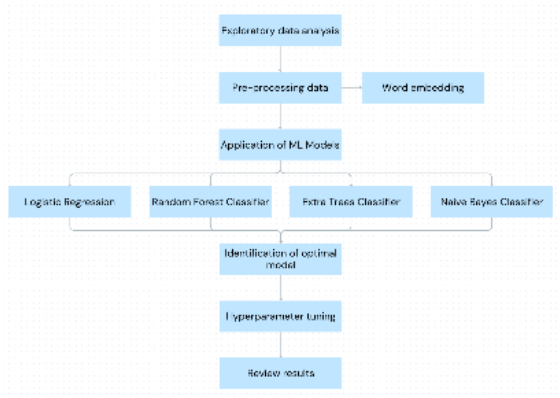


Figure 3: Flowchart of model development

4 Results and Discussion

Results

As our dataset is balanced with equal amounts of positive and negative classes, with no emphasis on the correctness of any class, accuracy remains the optimum metric to evaluate our models. We have also gathered metrics like precision, recall and F1-score to assist our evaluations, defined as following:

$$\textit{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$
$$\textit{Precision} = TP/(TP+FP)$$
$$\textit{Recall} = TP/(TP+FN)$$
$$\textit{F1-Score} = 2 \text{ x } (\textit{Precision x Recall})/(\textit{Precision} + \textit{Recall})$$

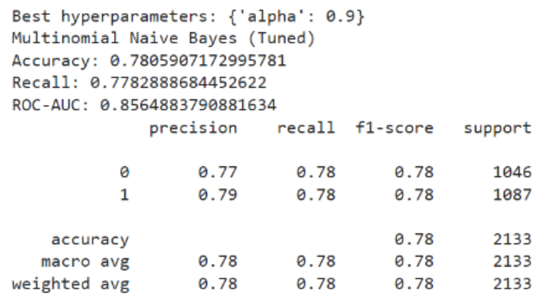
A summary of the accuracy of these models are described in the following table, grouped by the type of embedding used (Word2Vec, TF-IDF).

Model Used	Accuracy	
	WORD2VEC	TF-IDF
Logistic Regression	0.584622597	0.746835443
Random Forest	0.552742616	0.695733708
Extra Trees	0.548992030	0.721987811
Naive Bayes	0.554149086	0.779653071
WordBlob*	0.643935841	
VADER*	0.649188631	

*WordBlob and VADER have their set-in embeddings. The accuracy value of the corresponding row represents the model's accuracy.

Table 4: Model accuracy using different embedding

According to the statistics gathered, we conclude Naive Bayes as the most optimal ML model, coupled with TF-IDF word embeddings. The Naïve Bayes model is further optimized by hyperparameter tuning, particularly by altering the value of alpha, a parameter that dictates the distribution of the probability of zero counts in the model. The hyperparameter tuning is implemented via grid search with cross-validation testing, which is introduced via the module "sklearn.model_selection". We deduce an optimum alpha value of 0.9, which increases the accuracy to around 0.78059, a minute increase in performance. The results of the model after tuning are described in detail by the classification report below.



```
Best hyperparameters: {'alpha': 0.9}
Multinomial Naive Bayes (Tuned)
Accuracy: 0.7805907172995781
Recall: 0.7782888684452622
ROC-AUC: 0.8564883790881634
              precision    recall  f1-score   support

           0       0.77      0.78      0.78      1046
           1       0.79      0.78      0.78      1087

    accuracy                           0.78      2133
   macro avg       0.78      0.78      0.78      2133
weighted avg       0.78      0.78      0.78      2133
```

Figure 5: Classification report of the optimal model

The number of negative and positive instances are similar, with almost identical metrics of precision, recall and F1-score, indicating the model's balanced ability in predicting both positive and negative sentiments.

Discussion

We identify Naive Bayes / TF-IDF as the optimum combination for the given dataset. However, both TF-IDF and Naive Bayes hold certain assumptions on the dataset, which may not be true in typical lexical patterns. For instance, TF-IDF disregards the order in which words appear in a sentence, which may be important in determining sentiments. Naive Bayes assumes independence between words used in the sentence; however, there are certain words that are used in tandem as pairs, such as "back and forth", which is not captured in the algorithm.

Yet despite such limitations, the combination outperformed other complex models and algorithms that take these nuances into account, especially Word2Vec word embedding, which has a significant difference from TF-IDF. We explain such discrepancies in terms of variance-bias trade-off. Most of the sentences in movie reviews are around ten to thirty words, such that single words are sufficient predictors for sentiment in most of the cases. Word2Vec, considering multiple relationships between words, may end up training a model with too high variance, thus overfitting the data instead.

We have also tested our data with the two models introduced at the start of the paper, WordBlob and VADER, which have open-source Python libraries. Surprisingly, our model performed considerably better than the two models. This could be attributed to the reason that these libraries are trained on social media lexicon, which may not be consistent with language patterns and behaviors in movie reviews. This also highlights the specificity of models, especially NLP, to the dataset used for accurate classification.

Limitations and Suggestions

As mentioned in the previous section, our combinations have certain intrinsic limitations. We output sentences that are predicted wrongly and categorize them into three categories:

- Sarcasms: Words are used to convey opposite meanings rather than its actual meaning. *E.g.* "He looks funny" can also be negative.
- Negation: Words are used with negation terms, such as "not" or "never" that change the sentiments of the words.
- Foreign Language: Words from a foreign language.

For sentiment analysis of foreign language sentences that require a new word corpus, we suggest using multiple models, each trained on different languages, to analyze the sentiment of sentences of different languages. This would require sorting the datasets by languages used by the sentences then using the respective models to predict their sentiments.

The first two issues are intrinsically tied to TF-IDF embedding, which only captures word frequencies but not the word order. To overcome the issues, it may be worthwhile to train our model on an augmented dataset that contains similar amounts of typical and sentences that contain sarcasm or negation. Alternatively, an altered TF-IDF embedding that captures certain word order may also help identify sentences of these categories.

References

Ali, K., and Pazzani, M. 1996. Error reduction through learning multiple descriptions. *Machine Learning*, 24:3, 173-206. https://doi.org/10.1007/s10994-006-6226-1

Aljedaani, Wajdi, et al. 2022. “Sentiment Analysis on Twitter Data Integrating TextBlob and Deep Learning Models: The Case of US Airline Industry.” *Knowledge-Based Systems*, vol. 255. https://doi.org/10.1016/j.knosys.2022.109780.

Benkendorf, D. J., and Hawkins, C. P. 2020. Effects of sample size and network depth on a deep learning approach to species distribution modeling. *Ecological Informatics*. https://doi.org/10.1016/j.ecoinf.2020.101137

Gunasekaran, K. P. 2023. Exploring Sentiment Analysis Techniques in Natural Language Processing: A Comprehensive Review. *arXiv (Cornell University).* https://doi.org/10.48550/arXiv.2305.14842

Hutto, C., and Eric Gilbert. 2014. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media,* vol. 8, no. 1, pp.216–225, https://doi.org/10.1609/icwsm.v8i1.14550.

J. J., Park, S. Chen, and R. K., Choo (Eds.), *Advanced Multimedia and Ubiquitous Engineering* (pp. 347-352). Springer Singapore. https://doi.org/10.1007/978-981-10-5041-1_57

Khan, M., and Srivastava, A. 2024. A Review on Sentiment Analysis of Twitter Data Using Machine Learning Techniques. International Journal of Engineering and Management Research, 14(1), 186-195. https://doi.org/10.5281/zenodo.10791471.

Kim, S.W., and Gill, J.M. 2019. Research Paper Classification Systems Based on TF-IDF and LDA Schemes. Human-Centric Computing and Information Sciences, 9(1). https://doi.org/10.1186/s13673-019-0192-7.

Loh, W. Y. 2011. Classification and Regression Trees. *WIREs Data Mining and Knowledge Discovery*, 1(1), 14-23. https://doi.org/10.1002/widm.8.

Mikolov, T., Chen K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/arXiv.1301.3781

Nick, T.G., Campbell, K.M. 2007. Logistic Regression. In: Ambrosius, W.T. (Eds.) *Topics in Biostatistics. Methods in Molecular Biology*, vol 404. Humana Press. https://doi.org/10.1007/978-1-59745-530-5_14

Poornima A., and Priya, K. S. 2020. A Comparative Sentiment Analysis of Sentence Embedding Using Machine Learning Techniques. *6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India*. https://doi.org/10.1109/ICACCS48705.2020.9074312.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206–215. https://doi-org /10.1038/s42256-019-0048-x.

Sharou, K. A.; Li, Z.; Specia L. *Towards a Better Understanding of Noise in Natural Language Processing*. https://doi.org/10.26615/978-954-452-072-4_007

Wickramasinghe, I., and Kalutarage, H. 2020. Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation. *Soft Computing*, 25(3), 2277–2293. https://doi.org/10.1007/s00500-020-05297-6

Xu, S., Li Y., and Wang, Z. 2017. Bayesian Multinomial Naïve Bayes Classifier to Text Classification. *Journal of Information Science*, 1(12).