IT1244

Sentiment Analysis For Large Movie Review Dataset

Group 18: Quek Yu Chern (A0208537B), Tu Zhehao (A0200130M),

Mao Kaiwen (A0233965W), Lee Pak Shuang (A0235228H), Teoh Sean Keat (A0239700J)

Abstract

Sentiment Analysis makes use of Natural Language Processing (NLP) to extract emotional sentiments from subjective information in the given corpus. In this project, we discuss several well known methods commonly used in sentiment analysis.

1. Introduction

Sentiment analysis, otherwise known as opinion mining, is a challenging problem in the field of NLP. It is an extremely powerful tool that empowers managers with the means to better understand their business. Some examples include the reception of products, customer satisfaction as well as customer loyalty. Understanding the sentiment of consumers allows for better data driven decision making.

The data used for this project focuses on binary sentiment classification and consists of 50 000 movie reviews in separate text files. Preliminary Exploratory Data Analysis (EDA) showed that the dataset was well-balanced, with an equal number of positively and negatively classified data. Additionally, each movie had a maximum of 30 reviews and only reviews with 1-4 stars and 6-10 stars were included, implying that the dataset was highly polar (either positive or negative sentiments with little ambiguity). As such, an equal split of 25 000 data points was chosen for both training and testing.

In this project, due to the simplicity of the task, techniques involving complex deep learning (DL) networks and pre-trained models used in complex NLP tasks like dependency parsing or aspect based sentiment analysis were avoided. Simpler models such as logistic regression and SVM, which are deterministic models, naive bayes, a probabilistic model, as well as a shallow multi-layer perceptron (MLP) neural network.

2. Data Preprocessing

As the raw data cannot be used in supervised learning, we had to convert it into a numerical representation using feature encoding. Data cleaning and dimensionality reduction was also done to improve training efficiency.

2.1 Noise Removal

The raw data was in html format and line breaks (‘br /’) appeared in almost all the review instances. This provided no value to the prediction of the sentiment and was removed to reduce noise during training.

2.2 Lowercase

All words were converted to lowercase to reduce the dimension of our feature matrix. In some reviews, words were capitalised for emphasis but these were outliers and only constitute a very small percentage. Additionally, the semantic meaning of these words remained the same after capitalization (e.g., “TERRIBLE ACTING”) in most cases.

2.2 Punctuation

Most of the punctuations add little to no value to the prediction. Some however, like “!” and “:)”, had semantic meaning. Thus, empirical base model performances using different punctuation preprocessing were compared and it was found that removing punctuation led to better performance generally.

2.2 Stop word Removal

Due to the short length of some reviews, it was possible for stop words to carry important information that were necessary for prediction. In fact, there was a consistent dip in accuracy and f1 score across all models when stop words were removed. Hence, they were not discarded.

2.3 Lemmatization

Lemmatization changes all grammatical forms of a word to its base word to decrease the overall number of features. For example, the sentence “I was amazed by the film I watched!” becomes “I be amaze by the film I watch” after lemmatization.

Although lemmatization resulted in a slight decrease in accuracy, the mild loss of information is outweighed by the significant decrease in training time.

2.4 Stemming

Stemming is similar to lemmatization but truncates all words to a predefined set of words. Using the same example, “I was amazed by the film I watched!” becomes “i wa amaz by the film i watch”.

Stemming was implemented as another form of dimensionality reduction and was tested alongside lemmatization to compare model performances.

3. Feature Engineering

Two methods were used to convert the preprocessed text corpus into a numerical representation. Namely, Bag Of Words (BoW) and term frequency inverse document frequency (Tf-idf).

BoW counts the number of occurrences of each word in a corpus. In essence, each index of the vector represents a unique word or symbol. The respective word counts will be numerated while the rest will be zero. *CountVectorizer* from sklearn package was used to implement BoW feature extraction.

Tf-idf is similar to BoW except the term frequency is divided by the inverse document frequency: the more common the word, the lower the rank. Since stop word removal was not performed, Tf-idf is especially useful as it

is able to reduce the impact of stop words on our model. In this project, sklearn's *tfidfVectorizer* was used to implement Tf-idf feature extraction.

N-grams were also investigated as they are useful when word sequences matter (Srinidhi, 2019). Since the corpus is large, vocabulary size truncation had to be performed to minimise the exponential growth of the feature vector dimensions with the addition of bigrams and trigrams.

4. Discussion

4.1 Results

We implemented the four aforementioned models which were quick to train on relatively large feature spaces without compromising on the robustness of the results. The results of all models are shown in Table 1.

Table 1. Model Performances*

Models	BoW		Tf-idf	
	F1 Score	Accuracy	F1 Score	Accuracy
Logistic Regression	0.8711	0.8718	0.8946	0.8946
SVM	0.8628	0.8628	0.8949	0.8948
Naive Bayes	0.8503	0.8502	0.8640	0.8640
MLP	0.8832	0.8832	0.8616	0.8616

* Best results were used, pre-processing varied slightly across models

From Table 1, it is implied that Tf-idf is better in most models as the encoding method. This result is within expectations as Tf-idf factors in the frequency of the word both within each instance, and the entire text data. An interesting observation was that apart from Logistic Regression, all other models performed best without lemmatization or stemming entirely.

Upon adding bigrams and trigrams, the input feature vector got so large that we had to arbitrarily truncate off all features except the top 10 000 in order to minimise the training time. Nonetheless, models which used the addition of bigrams and trigrams outperformed all unigram models.

When looking into misclassified instances, we found that names have the potential to have both negative and positive sentiments. For example, the frequent mentioning of actors' names in positive reviews causes a general positive sentiment attached to them. Consequently, any review containing an actor's name may result in a very high probability of a positive prediction regardless.

Another issue was the presence of sarcastic reviews where the reviewer would ironically state how well an actor/movie had performed to hyperbolize their poor showing (e.g., "The plot was sooo interesting I fell asleep halfway through"). Although rare, where sarcasm occurs, there was an inconsistency in sentiment prediction

It is worth mentioning that due to the simplicity of binary classification, DL models may overfit easily and result in worse performances in general. In fact, all of our models outperformed implementations using Bi-LSTMs[1] and transformers[2] on the same dataset. The high F1 scores and accuracies across all models also implies that simple models are sufficient to make good sentiment predictions on this particular dataset.

4.2 Limitations

Both BoW and Tf-idf are sparse matrices (Chauhan, 2022). That is, after feature extraction, most (>50%) of the entries will be zero. This results in inefficient space allocation as memory is required for each zero value, but these values do not contribute any information to the model.

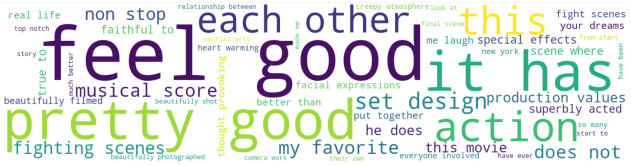
Additionally, both BoW and Tf-idf are only useful as lexical level features (i.e., word length) and are unable to capture semantics such as tone and topic (Scott, 2020). Thus, little insight can be derived solely using such models.

5. Extension: Topic Modelling

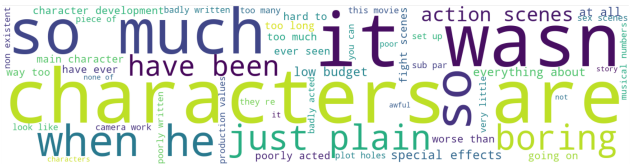
Our models can accurately predict a review's sentiment, but do not offer much insight into the attributes of a movie which compel positive or negative reviews due to the "Black-Box" nature of Machine Learning. We aimed to develop an Interpretable AI that would identify what aspects of a movie warranted a positive or negative review.

To do this, Topic Modelling was conducted on the reviews to determine the dominant topics. The reviews were first run through a Question-Answering model based on MiniLM, asking the question "What was the best aspect of this movie?" and "What was the worst aspect of this movie?", respectively. The Top2Vec library (Angelov, 2020) was used to model the topics: The extracted answers were then embedded using the Doc2Vec model, reduced in dimensionality using Uniform Manifold Approximation and Projection (UMAP), and finally clustered using Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Taking the centroids of the clusters and identifying the vectors nearest to each centroid, we arrive at a set of words that are representative of each cluster, shown below.

Sample of Words Related to Good Movies



Sample of Words Related to Bad Movies



6. Conclusion

With proper data pre-processing, even simple models such as logistic regression are sufficient to have high accuracy on basic sentimental analysis tasks. As such, these models are preferable if only surface level analysis is required.

Nonetheless, as the depth of analysis increases, higher complexity models might be required to learn high level features. For future works, dense word embedding can be investigated using DL architecture. This allows us to preserve syntactic and semantic information which can improve both model robustness as well as derived insights.

References

Amit Chauhan. (2022). *Difference between Bag of Words (BOW) and TF-IDF in NLP with Python*. Towards Ai. Retrieved from: https://pub.towardsai.net/difference-between-bag-of-words-bow-and-tf-idf-in-nlp-with-python-97d3e75a9fd

Dimo Angelov. (2020). *Top2Vec: Distributed Representations of Topics*. Retrieved form: https://arxiv.org/pdf/2008.09470.pdf

Jake Scott. (2020). *What's in a word?* Towards Data Science. Retrieved from: https://towardsdatascience.com/whats-in-a-word-da7373a8ccb#:~:text=TL%3BDR%3A%20Term%20Frequency%2D,resistant%20on%20the%20intensive%20margin.

Sunny Srinidhi. (2019). *Understanding Word N-grams and N-gram Probability in Natural Language Processing*. Towards Data Science. Retrieved from: https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058

Appendix

Table A1. Results of different preprocessing methods applied to the models*^

Model	Punctuation Removal	Lower case	Stopword removal	Stemming	Lemmatization	Domain Specific Vocab	Bigrams	Trigrams	BOW (Accuracy)	BoW (F1 Score)	Tf-idf (Accuracy)	Tf-idf (F1 Score)
Logistic Regression									0.8574	0.8574	0.8841	0.8841
	1	1							0.8605	0.8605	0.8865	0.8865
	1	1	1						0.8556	0.8556	0.8847	0.8847
	1	1		1					0.8571	0.8571	0.8825	0.8825
	1	1			1				0.8595	0.8595	0.8841	0.8841
	1	1			1	1			0.8520	0.8520	0.8810	0.8810
	1	1			1		1		**0.8711**	**0.8718**	0.8934	0.8934
	1	1			1		1	1	0.8698	0.8698	**0.8946**	**0.8946**
SVM									0.8461	0.8458	0.8831	0.883
	1	1							0.8408	0.8400	0.8863	0.8863
	1	1	1						0.8443	0.8443	0.8836	0.8836
	1	1		1					0.8506	0.8505	0.8832	0.8831
	1	1			1				0.8544	0.8544	0.8795	0.8794
	1	1				1			0.8394	0.8387	0.8824	0.8823
	1	1					1		0.8637	0.8637	0.8936	0.8936
	1	1					1	1	0.8586	0.8581	**0.8949**	**0.8948**
	1	1		1			1	1	**0.8628**	**0.8628**	0.8938	0.8937
	1	1			1		1	1	0.8524	0.8519	0.8936	0.8936
Multinimial NB									0.8212	0.8208	0.8370	0.8368
	1	1							0.8246	0.8243	0.8418	0.8416
	1	1	1						0.839	0.8388	0.8434	0.8432
	1	1	1	1					0.8247	0.8243	0.8309	0.8306
	1	1	1		1				0.8349	0.8347	0.8401	0.8400
	1	1				1			0.8218	0.8214	0.8390	0.8389
	1	1	1			1			0.8362	0.8360	0.8410	0.8408
	1	1					1		0.8445	0.8445	**0.8640**	**0.8640**
	1	1					1	1	0.8437	0.8437	0.8630	0.8630
	1	1	1				1	1	**0.8503**	**0.8502**	0.8566	0.8566
MLP (100,)									0.8576	0.8575	0.8356	0.8355
	1	1							0.8626	0.8626	0.8394	0.8393
	1	1	1						0.8496	0.8495	0.8342	0.8341
	1	1		1					0.8432	0.8431	0.8232	0.8231
	1	1			1				0.8577	0.8577	0.8366	0.8366
	1	1				1			0.8578	0.8577	0.8335	0.8335
	1	1					1		0.8790	0.8790	0.8610	0.8610
	1	1					1	1	**0.8832**	**0.8832**	0.8609	0.8609
	1	1			1		1	1	0.8813	0.8813	**0.8616**	**0.8616**
MLP (250,)	1	1					1	1	0.8875	0.8875	-	-
MLP (500,)	1	1					1	1	0.8896	0.8896	-	-
MLP (1000,)	1	1					1	1	0.8899	0.8899	-	-
MLP (2500,)	1	1					1	1	0.8904	0.8904	0.8815	0.8815
MLP (100,100)	1	1					1	1	0.8763	0.8763	-	-
MLP (250,100)	1	1					1	1	0.8782	0.8782	-	-
MLP (500,250)	1	1					1	1	0.8792	0.8792	-	-

* In this table, a value of 1 represents that the preprocessing was used for the model; the first entry of each model is the control

^ Best results cited in the main text have been bolded