IMDB SENTIMENT ANALYSIS

LUO, ROBIN (260851506), ROUSSEAU, MARC-ANDRE (260089646), AND XU BIDE, (260711367)

Abstract. We investigated the predictive power of several well known algorithms to perform sentiment analysis on imdb movie reviews. Using a linear support vector machine model trained on a dataset of 25000 movies reviews, we were successful in processing the text of a review and predicting whether the review was positive or negative with an accuracy of 91.786 percent on a dataset of 25000 reviews with the true sentiments withheld. After extensive analysis of various models, we concluded that support vector machine slightly outperformed linear regression which significantly outperformed the Naive Bayes implementation. The predictive power of a simple support vector machine model including tf-idf, minimal text preprocessing and tetragrams performed extremely well which made it very difficult to determine other features which may have significantly improved predictive power in a less parametrized model.

1. Introduction

The ubiquity of social networks is no longer a budding phenomenon and is part of the reality in which we now find ourselves. Many popular sites have included ways for users to share their opinions on a variety of topics and therefore the ability to mine through these posts and determine how users feel about the things being discussed is extremely useful for businesses. Knowing that a user or group of users desire something or find it appealing creates a market of opportunity for companies in search of low risk opportunities to expand their business operations. This is one of the most alluring aspects of machine learning. While the discovery of many of the mathematical underpinnings of machine learning have been discovered for quite some time, it is the rise in computing power which has given us the ability to use these techniques in otherwise intractable situations. For example gradient descent was invented by the prolific mathematician Augustin-Louis Cauchy in 1847. For our project, we were given 12500 positive and 12500 negative reviews to train our algorithm and another 25000 to test our code and submit our best guess as to the correct labeling of the test reviews as either positive or negative. Our best model which used a linear SVM (0.918 accuracy), was much better than Naive Bayes (0.83).

2. Related Work

Machine learning and sentiment analysis are hot topics of research with many machine learning conferences having several talks on the topic. For example, Twitter has been releasing datasets to be mined for things like whether a piece of task is positive, negative or neutral. Recently, a group of researchers extended this problem to five classification categories and added arabic language content (Rosenthal, 2017). Many teams submitted ML proposals to classify the twitter posts and the top performing groups used deep neural networks (DNNs) (Rosenthal, 2017). In addition, out of the top 10 submissions, the second most successful approach to DNNs involved the use of SVMs which is consistent with our best performing model for this project. A more directly related work involved taking into account sentence negations (Das, 2018). In this paper, the authors have decided to use a shortcut for negation by negating the word immediately following a negation. One example of this method would be to take the sentence "I am not happy" which gets converted to "I am not_happy". The benefit of this is that by changing only one word, they are able to change the meaning of the entire sentence (Das, 2018). Our team implemented this feature but it did not have a significant impact on the score after verification by cross-validation and kaggle submission.

3. Dataset and Setup

Our training dataset consisted of a list of 25000 reviews properly marked as positive or negative for us to train. We also had another 25000 data points to test out our model. The data consist of the content of the reviews and both the training and test sets needed to have their data preprocessed to be able to be consumed by the python libraries. For the purpose of training and validation, we separated our data into 80% training and 20% validation.

Date: February 23, 2019.

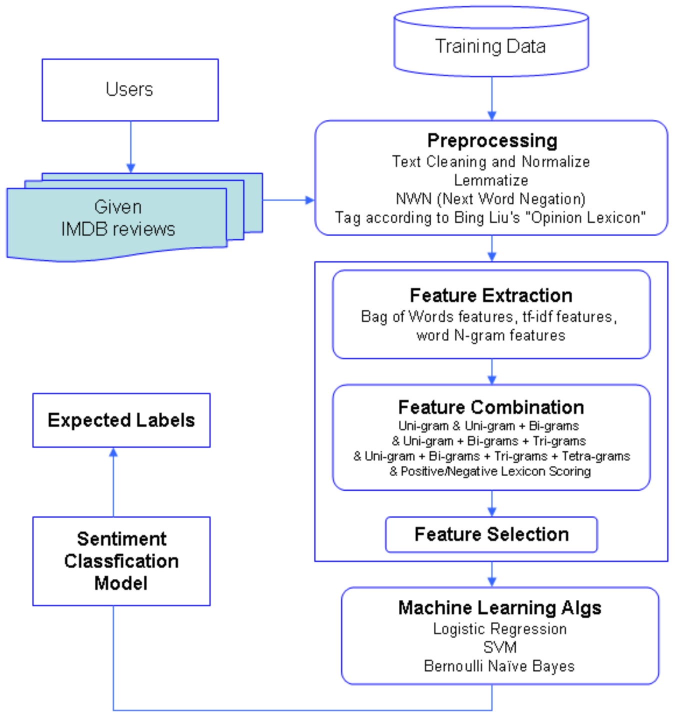

Training Data
Users
Given IMDB reviews
Preprocessing
Text Cleaning and Normalize
Lemmatize
NWN (Next Word Negation)
Tag according to Bing Liu's "Opinion Lexicon"
Feature Extraction
Bag of Words features, tf-idf features, word N-gram features
Feature Combination
Uni-gram & Uni-gram + Bi-grams
& Uni-gram + Bi-grams + Tri-grams
& Uni-gram + Bi-grams + Tri-grams + Tetra-grams
& Positive/Negative Lexicon Scoring
Feature Selection
Expected Labels
Sentiment Classfication Model
Machine Learning Algs
Logistic Regression
SVM
Bernoulli Naïve Bayes

Figure 1. **Flow chart of our proposed approach** The elements that were tested but not ultimately included were the NWN, opinion lexicon tagging as well as the positive and negative lexicon scoring

4. Proposed Approach

The approach we considered is summarized in the following flow chart. Not all of the elements were eventually considered in our final report since they did not provide any substantial performance benefit above and beyond that which our current models were able to meet.

4.1. Preprocessing.

For the preprocessing, we considered several models including a raw text input model, a model where the text was cleaned to only include letter characters (LC), a model where the text was processed through a normalization library and the words were separated from the punctuation symbols (SWP), a model with lemmatization (LZ) and a model with next word negation (NWN) from the previously mentioned paper. For all models with the exception of the raw data, the text was converted to lowercase characters. Following the initial preprocessing, the data was subsequently tokenized. For the normalization model, we removed any characters for which the normalization library could not process. This was usually due to unicode characters which could not be mapped to equivalent ascii characters.

4.2. Feature, hyperparamter and model selection.

In addition to the features generated by the preprocessing, we used tf-idf and bag of words (BOW) in our pipelines for feature selection. We implemented SVM, Laplace-smoothed Bernoulli Naive Bayes (NB) and logistic regression (LR) models. To determine the hyperparameters and for regularization as well as the feature selection, we used a grid-based search algorithm with a cross-validation parameter of 5. For our BOW feature, we tested out many different values for possible n-gram size and settled with 1-4 after testing out several different options. For the SVM model, we tried values for the penalty parameter ”C” and decided to use the default value of 1.0. In addition , once our model was able to associate good or bad to a particular set of words, we used these word to sentiment mappings to analyse subsentences separated by punctuation. For each bad or good word found, we added 1 or -1 to a counter and then classified the value for the overall text to the sum of all the individual good/bad words found in the text. If purely estimate the output class based on this calculated sentiment score, such that if the overall score is larger than or equal to 1, then predict positive. Or else the overall score is less than or equal to -1, then it predicts a negative review. For the case that this overall score equals 0, then use the original text for prediction with tfidf. This method could result in an accuracy of roughly 75%, as shown in Table 1, line 35. Furthermore, we used a normalization library to standardize the feature vectors so that they had unit length. In terms of the theoretical underpinnings of the model selection, we used a data-driven approach rather than a conceptual one and so there was little discussion about which model was more appropriate and instead we opted to go with the model that had the best predictive power.

5. Results

The task of deciding on a final model based on the performance is quite easy since we only had to code the model and then pick whichever model worked the best, however a discussion of the relative performance of the models is difficult due to how close the top models performed as well as the sheer quantity of possible adjustments. In general, tf-idf outperformed bag of words, SVM outperformed LR and a model allowing for tetragrams or trigrams outperformed models with only unigrams or bigrams. In most cases, both trigrams and tetragrams has similar results.

5.1. **NB vs LR vs SVM.** The performance of the Naive Bayes was significantly better than a random guess approach but still failed to come close to the performance of the logistic regression or the support vector machine approaches. Our top performing model on cross-validation was a SVM model, in line 32 of the table, but this did not extend to kaggle submissions. While another model for which is also a SVM model in line 34 slightly get one of our best scores in kaggle.

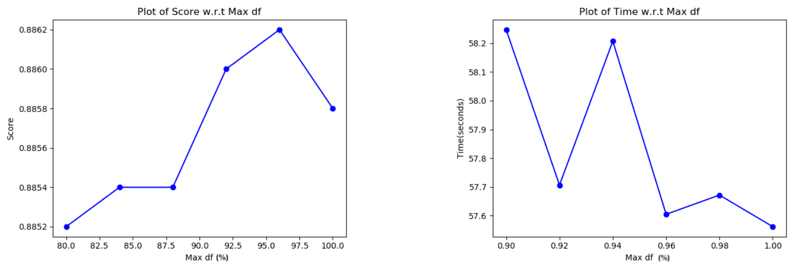
5.2. **Bag of words vs tf-idf.** The performance of SVM and LR using both BOW and TF-IDF were similar with TF-IDF being slightly better than BOW according to the parameters we chose for min-df and max-df. For our model, we used a min_df of 2 and a max_df of 1.0 meaning that words that appeared in at least 2 of the documents were considered and no constraint was placed on the maximum frequency that a word could occur. In addition to using these models, we had originally also included stop_words but these did not show any additional gains above and beyond tf-idf and therefore it would seem as though the idf aspect of the tf-idf implementation may have already been incorporating the information contained in stop_words by essentially removing those words which were used too frequently by lowering the respective weights. Looking at the values in Table 1, we can see that for models where BOW and TF-IDF were used, the latter usually performed slightly better. A strict comparison of these features is difficult due to the many possible parametrizations of the BOW implementation. Figure 1 shows how the max_df affects the score and the difference in the score between the BOW and tf-idf be partially due to the inclusion of those common words which appear in almost every paragraph of english (”in”,”the”, etc...).

5.3. **N-grams.** We considered many different possibilities for n-grams but ultimately we settled on using (1-4)-grams even though the gain in performance was quite small between limiting ourselves to trigrams and including tetragrams. The logistic regression model with td-idf showed that the value in increasing above tetragrams was nonexistent. While the increase in score between the trigram and tetragram models was under 1 percent for most models, the increase seemed consistent across almost all models considered and was therefore included. The additional value added in considering fourth level n-grams is probably due to common four-word expressions in the english language which convey sentiment. The low increase in performance is quite simply due to the low frequency of occurrence of these expressions overall for the amount of data we had.

Table 1. Runtime and F1 performance analysis of various models

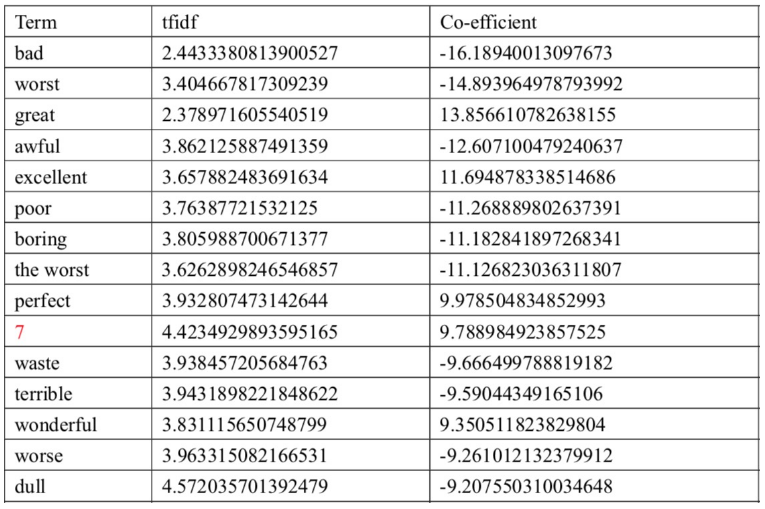
Number	Model	F1	#feat.	f. runtime (s)	p. runtime (s)
1	Bernoulli NB + BOW	0.84879	67708	240	1020
2	Bernoulli NB, LNs (letters and numbers), BOW, 1gram	0.85250	69035	180	960
3	LR, BOW, LN, 1-gram	0.88780	69035	17	3
4	LR,LCs, 1-gram	0.88561	27254	10	2
5	LR, BOW, LCs, (1-4)-gram	0.88961	104211	106	7
6	LR, tf-idf, LCs, (1-4)-gram	0.89481	104211	95	6
7	LR, tf-idf, raw, 1-gram	0.88541	27745	29	4
8	LR, BOW, raw, (1-2)-gram	0.89521	166041	54	9
9	LR, BOW, raw, (1-3)-gram	0.89601	290118	114	14
10	LR, BOW, raw, (1-4)-gram	0.89521	342886	230	230
11	LR, tf-idf, raw, 1-gram	0.89443	27745	20	6
12	LR, tf-idf, raw, (1-2)-gram	0.91027	166041	55	8
13	LR, tf-idf, raw, (1-3)-gram	0.91175	290118	110	10
14	LR, tf-idf, raw, (1-4)-gram	0.91199	342886	199	20
15	LR, tf-idf, raw, (1-5)-gram	0.91203	358250	211	13
16	LR, tf-idf, cleanup, (1-3)-gram	0.91123	284493	116	12
17	LR, tf-idf, cleanup, (1-4)-gram	0.91083	334013	169	18
18	LR, tf-idf, cleanup, SWP, (1-3)-gram	0.91135	284955	117	10
19	LR, tf-idf, cleanup, SWP, (1-4)-gram	0.91099	333673	166	20
20	LR, tf-idf, cleanup, lemmatize, (1-3)-gram	0.91019	283063	108	11
21	LR, tf-idf, cleanup, lemmatize, (1-4)-gram	0.91083	334089	178	12
22	LR, tf-idf, cleanup, NWN, (1-3)-gram	0.91067	264522	89	9
23	LR, tf-idf, cleanup, NWN, (1-4)-gram	0.91119	297461	147	11
24	LR, tf-idf, cleanup, Lemmatize, SWP (1-3)-gram	0.91083	283682	85	8
25	LR, tf-idf, cleanup, Lemmatize, SWP (1-4)-gram	0.91095	333997	141	11
26	LR, tf-idf, cleanup, Lemmatize, SWP, NWN, (1-3)-gram	0.91131	263993	90	9
27	LR, tf-idf, cleanup, Lemmatize, SWP, NWN, (1-4)-gram	0.91127	298356	132	10
28	SVM, tf-idf, raw, (1-4)-gram	0.91239	1010594	167	13
29	SVM, tf-idf, cleanup, (1-4)-gram	0.91271	987640	155	15
30	SVM, tf-idf, cleanup, NWN, (1-4)-gram	0.91295	899187	156	12
31	SVM, tf-idf, cleanup, Lemmatize, (1-4)-gram	0.91271	987519	171	12
32	SVM, tf-idf, cleanup, SWP, (1-4)-gram	0.91307	988012	161	16
33	SVM, tf-idf, cleanup, Lemmatize, SWP, (1-4)-gram	0.91295	989022	162	15
34	SVM, tf-idf, cleanup, SWP, Lemmatize, NWN, (1-4)-gram	0.91251	901848	164	13
35	Sentiment count estimation, tf-idf, (1-4)-gram	0.75392	23297	8	1

Summary of selected models considered for our final submission. The runtimes are the fit and predict runtimes respectively. The final model performance was determined via a study of the effect of modifying the adjusting the model on the F1-scores. All runtimes with the exception of the Naive Bayes did not preclude extensive testing. A subset of the models under consideration have been presented.



(A) score vs max_df (B) total runtime vs max_df

FIGURE 2. **Performance analysis of the max_df parameter** In A, we can see the best prediction is obtained just shy of the 100 percent mark which might account for why our BOW model did slightly worse than the tf_idf model which would automatically account for the high frequency words. In (B), we see that above the 90 percent mark, there is little effect on the runtime which coud simply be due to the fact that there are few words which appear extremely frequently and therefore there is no discernable effect on runtime



Term	tfidf	Co-efficient
bad	2.4433380813900527	-16.18940013097673
worst	3.404667817309239	-14.893964978793992
great	2.378971605540519	13.856610782638155
awful	3.862125887491359	-12.607100479240637
excellent	3.657882483691634	11.694878338514686
poor	3.76387721532125	-11.268889802637391
boring	3.805988700671377	-11.182841897268341
the worst	3.6262898246546857	-11.126823036311807
perfect	3.932807473142644	9.978504834852993
7	4.4234929893595165	9.788984923857525
waste	3.938457205684763	-9.666499788819182
terrible	3.9431898221848622	-9.59044349165106
wonderful	3.831115650748799	9.350511823829804
worse	3.963315082166531	-9.261012132379912
dull	4.572035701392479	-9.207550310034648

FIGURE 3. **Sorted features after tf-idf and training to determine weights.** The number 7 appears to be very strongly associated with a positive review.

5.4. **Text processing.** The effect of processing and preparing the text was a contributor to performance as can be seen by comparing the values in Table 1. We can see that by limiting ourselves to letters exclusively, we actually lose some information and the score for the LC models suffered as a result. As an example, the model using LR, tf-idf and (1-4)-grams had a performance score of 0.895 for the letter character model, a score of 0.912 for the raw text and 0.911 for the model where the text was normalized and converted to lowercase. An increase in performance between the raw and cleaned data was not universally consistent which could be due to the cleaning removing some information that would otherwise have been useful. One of the most interesting results is that some of the numbers actually ranked quite high after td-idf weighing and feature weight calculations. The model which removed everything but letter characters would have removed this important feature and therefore lost some predictive power as a result.

5.5. **Lemmatization.** The non-lemmatized model seemed to perform consistently better than the lemmatized model, however the submission to kaggle suggested that there was in fact a gain in performance in using the lemmatized model. Lemmatization is a feature which reduces the information rather than increases it and therefore it is not surprising to see some drop in perfomance. Despite this, the idea behind lemmatization is that there are many words for which only the root of the word conveys information and therefore this should help reduce noise and regularize so as to better extend our model to unseen data.

5.6. **SWP and NWN.** The SWP and NWN models did not seem to provide significant improvements to our model after having already incorporated tf-idf, lemmatize, and a cleaning of the text. While it is true that our best performing model used SWP (Table 1), this result did not extend into the kaggle submission and could be due to random variance in the data. We had expected some gain in performance from the NWN and had also tried other forms of regularization of negation but no clear improvement was seen in our kaggle submissions or our cross-validation.

6. Discussion and Conclusion

6.1. **Not losing information when pre-processing.** As we began to work with the data, it became clear that while taking care to properly clean and process the data does provide some benefit, in the presence of powerful features like BOW and tf-idf, giving us a high accuracy, there is very little room for improvement in subtlely massaging the data to make it more suitable. In addition, preprocessing can actually remove important information as was seen in the table listing the highest weighted words showing the importance of the number 7. In trying to understand why this might be the case, we browsed metacritic.com to gain some insights into what an average score is and it turns out to be 63/100. On rotten tomatoes, a fresh review is anything above a 60 whereas a rotten review is anything 59 and below. It could be that having a user mention the number 7 as being related to how they have rated the movie or how they feel about a movie will more often or not be associated with a positive review since the a leading 7 would be a fair bit higher than the threshold to cross over into a bad review.

6.2. **Difficulty of finding optimization by adding custom features.** In implementing SWP and NWN, we were hoping to see some gains in performance but with a relatively small dataset compared to the number of features, it is difficult to see improvements after having implemented components like tf-idf, bag of words, lemmatize, etc... To be able to truly create more advanced features, we would either have to consider models that could break down sentence structure or alternatively we could consult someone with domain knowledge of the imdb reviews who could give us insights about how people write their reviews so we could add these to our models.

6.3. **Capitalization.** While our final model has all words converted to lowercase, it is important to note that there may be a loss of predictive power in doing so. To be more certain of this, we would have to have a better idea of where the reviews came from. If the reviews were submitted by users with little to no oversight, then you could expect people writing words in all CAPITAL LETTERS to emphasize a certain point. This is a potential avenue for further investigation.

6.4. **Domain Knowledge.** While some of us enjoy watching movies, none of the members of our group are experts at reading and analyzing movie reviews and so our analysis is inherently limited to that which someone with a small amount of domain knowledge and a fair bit of computing knowledge can achieve. In certain situations, having a well informed expert to help guide the machine learning scientist would not only have the potential of improving the performance of the model but it might also help in generating interesting questions of inference about the data.

7. Future Considerations

If we were to continue working with the dataset, we might want to consider using a neural network or consider using category theory to better mine sentence structure for important components. Work involving category theory and natural language processing has been around for many years. One example being recent work by Kartsaklis et al. (Kartsaklis, 2016).

8. Division of Work

- **Robin Luo**: Model fitting, analysis of the effect of hyperparameters
- **Marc-Andre Rousseau**: Literature research, TeXing, some minor coding.
- **Peter Xu**: Coding, feature selection and hyperparameter fitting.

9. References

1 Rosenthal, Sara, et al. SemEval-2017 task 4: Sentiment analysis in Twitter Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pp. 502518.

2 Das, Bijoyan. Chakraborty, Sarit. "An improved text sentiment classification model using tf-idf and next word negation" June, 2017, eprint arXiv:1806.06407

3 Kartsaklis, D., Sadrzadeh, M., Pulman, S., & Coecke, B. (2016). Reasoning about meaning in natural language with compact closed categories and Frobenius algebras. In J. Chubb, A. Eskandarian, & V. Harizanov (Eds.), Logic and Algebraic Structures in Quantum Computing (Lecture Notes in Logic, pp. 199-222). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139519687.011