Using tripadvisor reviews to predict sentiment

Mathias Grotøy Bjørgum
Business Administration / 2022
mathigb@stud.ntnu.no
https://github.com/MathiasBjorgum/TDT4310---Project

Abstract

This paper is a project report in *TDT4310, Intelligent Text Analytics and Language Understanding*. The project explores how different classifiers predict sentiment from tripAdvisor reviews. This kind of data could be extremely important to businesses that work in tourism, since it gives a unique insight in peoples feelings based on what they write. The results show that a votingClassifier can be quite effective in predicting if a review has given five stars or not, with an F1-score of $\approx 74\%$, although not better than a single Support Vector Classifier. Lemmatizing and stemming the text data seems to have little impact on the accuracy.

1 Introduction

There is a lot of unstructured data in the form of text available on the online. Being able to use this to predict the sentiment of the text could be useful to get the computer to make an informed decision, like a person would.

The reason for choosing Tripadvisor data is because the dataset was avaliable online Bansal (2018). The main goal of this task is not to directly predict sentiment of a tripadvisor review since the number of stars is displayed with the review, but rather to create models that can predict sentiment where stars are not present. Furthermore it could be interesting to see if models in this field can make star ratings obsolete, and the text alone could give companies all the information they need.

In this project I have chosen to only look at five stars as a positive sentiment, and all others as a negative. The reason for this is to see how well a model can predict if something is "perfect" or if it leaves some room for improvement. Another reason for this is that there is no clear definition of what is required to get a certain star-rating. I believe that five-stars might be the rating that is the least ambiguous and therefore gives the most trustworthy result.

The project tries to find out what kind of classifier that performs best on this type of text classification problem. It uses Accuracy and F1-score as evaluation metrics. The project also explores if stemming and lemmatizing have any impact on the models performances.

2 Background

To understand the significance of the report, i will first describe some terminology that is relevant for this topic.

2.1 Stemming and lemmatizing

Theese two concepts apply to language understanding, and is often used to make text more readable for the computer, and better to vectorize.

Stemming is removing affixes and suffixes from words. Lemmatizing on the other hand is restricting a word to its main part. For example would "gardening" be lemmatized to "to garden". If we use a stemmer, "gardener, garden, gardening" would all be stemmed to "garden" (Bengfort, 2018).

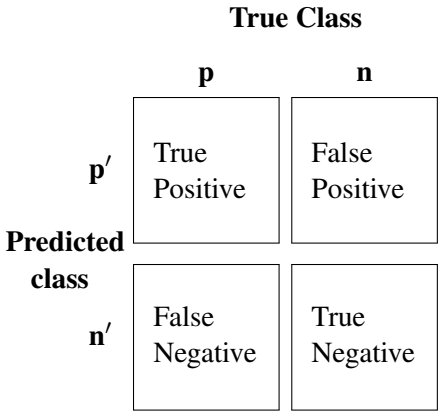

Figure 1: Confusion matrix illustration.

2.2 TF-IDF

TF-IDF is short for Term Frequency - Inverse Document Frequency. This is a method for vectorizing text into numbers for the computer to read. TF is derived from how often a given word appears in the corpus. IDF is then used to give the TF some context. This method takes into account that some words have i high relevance in certain topics (Bengfort, 2018).

2.3 Metrics

In the evaluation of my models i use some different evaluation metrics. To understand the metrics, and how we get them we must first introduce the confusion matrix. The matrix is illustrated in Figure 1. The matrix shows how we categorize a prediction given its true class. For example, if we predict something to be negative, but it is actually positive, the classification would be labeled as a False Negative (FN).

Precision and Recall. These measurements give the foundations for further calculations. Precision is a measurement of how many true positives we got from all predicted positives. Recall is a measurement of how many true positives we retrieved among all positives. The formulas are listed in Equation 1, 2 below.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

F1-score. This metric is used to calculate how effective a classification model is. It is a combined metric that takes into account both *Recall* and *Precision*. It is the harmonic mean between the two measurements. The calculation is shown in Equation 3.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3}$$

Accuracy. This metric gives the amount of correctly predicted classes. That is, both TP and TN. One disadvantage with this metric is that it performs rather poorly if the dataset is unbalanced. If 70% of the data belongs to one class, a model that predicts everything to be that class would get an *Accuracy* of 70%. The formula is shown in Equation 4.

$$accuracy = \frac{TP + TN}{P + N} \tag{4}$$

3 Related Work

One relevant article on this topic is Alaei et al. (2019). The article explores how Big Data can be used in the tourism domain. Multiple machine learning methods were applied, among them *SVM* and *Naïve Bayes*. The authors get *F1-Scores* in the range 0.6 - 0.9. The *accuracy* is in the high 0.8.

Another article, Chen et al. (2020), explores sentiment analysis on online travel review text. It has transformed the classification problem into a multi-classification problem, where it tries to predict review stars. The article emphasizes keyword extraction as something one should include. It also gets an F1-score above 0.9, witch is quite high.

Since i have retrieved the dataset from kaggle, there are other notebooks on the same topic. However they are not peer reviewed and are thus not that polite sources. I have looked some into the different solutions and they get accuracy scores around 0.9 on this dataset. Many of these solutions use deep learning and neural networks to get these scores.

4 Architecture

This section will explain the overall architecture of the model, and explain a bit about the dataset.

4.1 Dataset

The dataset itself is available publically on zenodo (Bansal, 2018), or on kaggle[1]. In Table 1 the dataset is visualized. I have included the raw ratings as well as the sentiment, where all ratings below 5 is considered negative. We can see that the dataset is fairly balanced when giving all ratings below 5 negative sentiment.

Table 1: Value counts for individual ratings ad the corresponding sentiment.

Rating	Count
5	9054
4	6039
3	2184
2	1793
1	1421
Sentiment	
Negative	11437
Positive	9054

4.2 Model

The model architecture is nothing special, and has been tested by many before. I have chosen to save my trained vectorizer to be able to vectorize on completely unseen data. Due to time limitations I have not been able to scrape my own data to test on, so i will have to rely on the data given in the dataset. The whole pipeline for the model is illustrated in Figure 2.

The first step in my pipeline is to gather the raw text. This is already done here, since the dataset is available*o*nline. The next step is to extract textual features. In this task this is limited to stemming or lemmatizing using `nltk.WordNetLemmatizer`[2] and stemming by `nltk.PorterStemmer`[3].

[1] https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews
[2] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[3] https://www.nltk.org/_modules/nltk/stem/porter.html

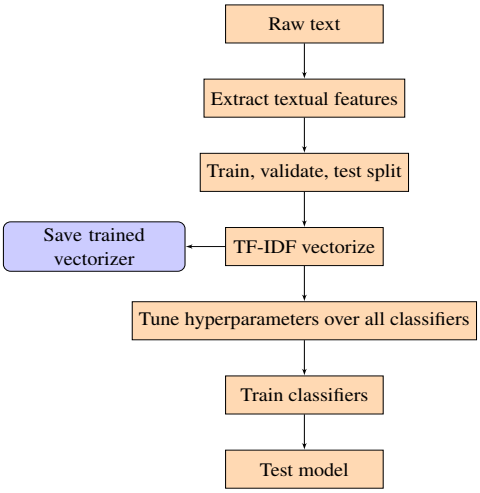

Figure 2: Model architecture.

Then we separate the data into a train and test set. With the validate set overlapping with the training set. After this we vectorize using a TF-IDF vvectorizer. In this step the fitted vectorizer gets saved for later use. The saved model will not be used in this project, but is put in place to make it easier to work on this later.

The hyperparemeters then gets tuned using the validation set, and we use these parameters to train the different classifiers. Finally we test the different models on unseen test data to gather evaluation metrics.

5 Experiments and Results

In this section i will present how my experiments are carried out, and what results i get from them. Due to randomization in different methods, the exact answers that i have gotten might differ from others. It should however not make any significant differences.

5.1 Experimental Setup

Different classifiers. I have used multiple different classifiers in this project. All of them (except the `DummyClassifier`) are then fed into a `VotingClassifier`[4] with voting set to "hard". All the classifiers are listed below:

- `DummyClassifier`[5]
- `SVC`[6]
- `KNeighborsClassifier`[7]
- `RandomForestClassifier`[8]
- `MultinomialNB`[9]

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

To get what hyperparameters to set i used `GridSearchCV`[10] with five folds, and used half of the training dataset. The resulting hyperparameters are listed in Table 2.

Table 2: Hyperparameters for the different classifiers. If nothing is specified, default values from scikit-learn were used.

Classifier - Parameter	Value
`SVC`	
C	1
Kernel	rbf
`KNeighborsClassifier`	
leaf_size	20
n_neighbors	7
weights	uniform
`RandomForestClassifier`	
n_estimators	120
criterion	entropy
`MultinomialNB`	
alpha	1

5.2 Experimental Results

Results from testing the trained models are given in Table 3. There are different evaluation metrics for whether the data has been stemmed, lemmatized or neither.

Table 3: Accuracy and F1-scores from testing the multiple classifiers given if they were lemmatized, stemmed or neither.

	No manipulation		Lemmatization		Stemming	
Classifier	Acc.	F1	Acc.	F1	Acc.	F1
`SVC`	.798	.776	.798	.776	.798	.776
`RandomForestClassifier`	.777	.741	.774	.741	.768	.730
`MultinomialNB`	.768	.741	.768	.741	.768	.741
`KNeighborsClassifier`	.694	.658	.694	.658	.694	.658
`DummyClassifier`	.543	.000	.543	.000	.543	.000
`VotingClassifier`	.779	.738	.777	.737	.778	.737

Based on this results, it does not seem like lemmatizing the text have any effect on the Accuracy and F1-score of the models. It performs worse on the `RandomForestClassifier`, witch might explain why the `VotingClassifier` also performs worse. Stemming does not apply any benefits either.

The model with neither Lemmatizaion nor Stemming gives the highest (or equal) F1-score and Accuracy for all of the models.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

6 Evaluation and Discussion

There has been many aspects of uncertainty when developing these models. I will try to include some choices and findings that were interesting.

6.1 The dataset

When i started looking for tasks i found this dataset on kaggle and thought it could make a good project. After working with it for a while i got to the conclution that it was a bit unbalanced, and found out that i would be stricter in what was considered a positive sentiment. I still think it is interesting to filter all non-perfect scores as negative.

My understanding is that a lot more people give tripAdvisor reviews if they are happy than if they are not. Some tasks have used the full 1-5 range when trying to predict sentiment, but I do not think this gives the correct picture, as there is a lot of debate around what requirements are in the different star-reviews.

6.2 Lemmatizing and stemming

I had predicted that either lemmatizing or stemming the data would have a positive effect on the evaluation metrics, but that was not the case. Since stemming and lemmatizing should give fewer total words it would be reasonable if the predictions were better. The reason for this might be that there are few words to begin with. Many people do not write full sentences when writing reviews, and thus lemmatizing and stemming might not give that much of an impact.

6.3 Classifiers

The `VotingClassifier` should in theory be better than one single classifier. We note that the `SVC` perform better than the `VotingClassifier`. The cause for this might be that Support Vector Classifiers overall is better than other classifiers. If we feed the `VotingClassifier` bad classifiers the result would also be bad. As we see in Table 3 the `KNeighborsClassifier` is the worst performer and then could drag the combined `VotingClassifier` down.

I am not shocked by my findings, since other attempts to classify on this dataset has yeilded similar resutls. The `KNeighborsClassifier` only hits the correct label around 15% times more often than the `DummyClassifier`, which labels everything to the most common label in the training set. This can indicate that it is hard to find the label while only considering what the nearest points have been classified to, and that we need more sophisticated models, like the `SVC`.

6.4 Other remarks

In hindsight I see that what i set out to find out might not be all that interesting. I wanted to see how well a classifier could predict whether a review was given five stars or not. Accuracy and F1-scores around 70-80% is acceptable, but the state-of-the-art is well above this, in the 90%. I could have extracted more textual features from the raw data, like word length or similar. If i had more time i think this project could be of more significance, since I learned a lot in the closing parts of the project. It is however important to also write reports that points out the flaws of an experiment, to be able to learn from them and make better experiments later.

7 Conclusion and Future Work

I started out this project to find out how well a classifier can predict perfect score on tripAdvisor data. With some tinkering with hyperparameters and using a TfidfVectorizer i have been able to get an F1-score of $\approx 74\%$ on a `VotingClassifier` and $\approx 78\%$ on a `Support Vector Classifier`.

This kind of result is in the area of earlier attempts at predicting sentiment on hotel review data. My approach is the only one i have found that has been this strict in what is considere positive. Interestingly lemmatizing and stemming the source data does not seem to have much impact on the classification results.

Future Work. Due to limitations in time and resources i was not able to perform any meaningful deep learning approaches to the problem. If this work were to be extended i would recommend exploring how those sort of methods could be implemented. I would also recommend exploring if using `word2vec` or `GloVe` to vectorice better.

References

Ali Reza Alaei, Susanne Becken, and Bela Stantic. Sentiment analysis in tourism: Capitalizing on big data. *Journal of travel research*, 58(2):175–191, 2019. ISSN 0047-2875.

Barkha Bansal. Tripadvisor hotel review dataset, April 2018. URL https://doi.org/10.5281/zenodo.1219899.

Benjamin Bengfort. *Applied text analysis with Python : enabling language-aware data products with machine learning*. O'Reilly, Beijing, 2018. ISBN 9781491963043.

Wen Chen, Zhiyun Xu, Xiaoyao Zheng, Qingying Yu, and Yonglong Luo. Research on sentiment classification of online travel review text. *Applied sciences*, 10(15):5275, 2020. ISSN 2076-3417.