Machine Learning Nanodegree Capstone project report

Kareem Ahmed Abdel-salam

Sentiment Classification using IMDB Dataset

October 10, 2018

Project Overview

Sentiment Classification is a common task in the field of Natural Language Processing, it is about extracting the sentiment of a person given a text they wrote as input.

Extracting sentiment from text can lead to build more intelligent systems acting according to those sentiments like chatbots or smart assistants.

Although this task is considered a complex task even for humans due to absence of facial expressions and voice tones in text but it is fairly much easier when given a paragraph of more of text written for the purpose of delivering sentiment such as the case with movie reviews.

The dataset was constructed back in 2011 by Andrew L. Mass et al. in the paper “Learning Word Vectors for Sentiment Analysis”, was a Kaggle challenge in 2016 and considered a Sentiment Analysis benchmark. Another Dataset used for this task is "Rotten Tomatoes" Dataset.

Problem Statement

Given the 50k labeled IMDB movie reviews in text format it is required to extract the sentiment and classify each review (from the 25k test set) to be either positive or negative, then compare the predicted results to the real labels and output the classification accuracy.

The problem is to be solved by extracting features out of blocks of text (reviews) and mapping feature values and combinations to either positive or negative sentiment classes through the usage of a couple of Supervised Machine Learning Algorithms (Logistic regression and Naive Bayes), Also some text preprocessing will be done to facilitate and improve quality of extracted features.

Evaluation Metrics

The metric used to evaluate the model is the Accuracy because it is a classification task and all we care about is whether or not the model can extract the sentiment from the reviews correctly.

So we determine how well the model is performing by finding the percentage of the correctly predicted labels.

Accuracy= (number of correct labels) / (total number of labels) * 100%

Datasets and inputs

The IMDB movie review dataset was first proposed by Maas et al.in ”Learning Word Vectors for Sentiment Analysis” as a benchmark for sentiment analysis.

The dataset consists of 100K IMDB movie reviews and each review has several sentences. The 100K reviews are divided into three datasets: 25K labeled training instances, 25K labeled test instances and 50K unlabeled training instances. Each review consists of several sentences of text in a file and has one label representing its sentiment: Positive or Negative. These labels are balanced in both the training and the test set.

The dataset can be obtained from : http://ai.stanford.edu/~amaas/data/sentiment/

I used the 50k labeled section of the dataset to train my supervised learning model.

Benchmark Model

A good benchmark to see whether a machine-learning solution works or not is random chance.

Given that the data is evenly split and the 25k test data is composed of 12.5 k positive samples and 12.5k negative samples then a random chance classifier will give a 50% accuracy, so my model must outperform this percentage in order to be effective.

Another historical benchmark to compare my solution to is the paper mentioned before and in the Credits section buy Andrew Mass : the highest classification accuracy was 88.89% on this dataset. So an efficient model will give close numbers or outperform it.

Analysis

By qualitative evaluation of some samples of data and reading Readme of the dataset, each review is a text file containing multiple sentences, the text contains upper and lower case letters, numbers, punctuation marks and HTML tags.

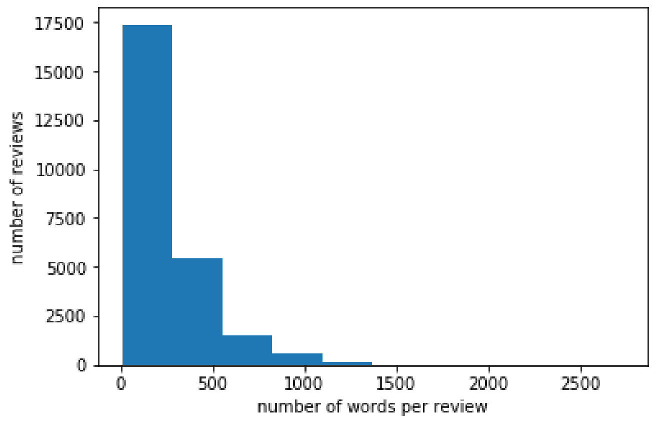
In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels.

Statistical Analysis:

By performing some statistical analysis on the reviews the maximum number of words in a review was 2730 words, minimum number of words : 11, mean: 264.5, median :198.0, standard deviation: 196.3.

From these numbers we can see that most reviews will probably be around the the median value and that the maximum value of 2730 is an outlier value.

Visualization:



By plotting this histogram where x-axis is the number of words per review and y-axis is the number of reviews we can clearly see that the data is right skewed and most reviews contain less than 600 words and only small portion of data contain more than that and it keeps decreasing. So this may indicate that the reviews with word count > a certain value are outliers and may confuse classifiers while they may be special cases (i.e. features obtained from them cannot be generalized to fit the whole distribution)

Samples

example of a positive sentiment training review:

“This movie has a special way of telling the story, at first i found it rather odd as it jumped through time and I had no idea whats happening.<br /><br />Anyway the story line was although simple, but still very real and touching. You met someone the first time, you fell in love completely, but broke up at last and promoted a deadly agony. Who hasn't go through this? but we will never forget this kind of pain in our life. <br /><br />I would say i am rather touched as two actor has shown great performance in showing the love between the characters. I just wish that the story could be a happy ending.”

example of a negative sentiment training review:

“I found this movie to be a great idea, that didn't deliver. It seems they found a way to build suspense, but couldn't stage their payoffs very well. In one case the police, are on the clock to find the hideout of the kidnappers. They painstakingly go from dentist to dentist to match a dental record. At the same time, the kidnapped man (Mason) escapes through the elevator shaft. After all the build up, the police arrive at the same time he gets free, which is very anti-climatic to say the least. There are also large narration scenes that take us "inside the thinking" of the terrorized husband and wife, which detracts from the suspense rather than adds to it. We are fully aware of their tension, and the voice-over is an insult and robs the viewer of any chance of a personal experience with the fear, as Hitchcock proved time and again, is far more effective. The greatest disappointment, is to sit through the whole movie, and the get the quick, rather bland ending. I mean it just..."ends" in a snore.”

example of a positive sentiment testing review:

“ If your expecting Jackass look somewhere else this an actual movie and for the budget well done the acting isnt top noth neither is the writing but the directing was there and so was the story definetly worth the rent and possibly the buy if you really enjoy it like i did. But for the person who just likes jackass rent it first.”

example of a negative sentiment testing review:

“ This show is just another bad comedy which will probably be cancelled after two seasons. It's not just that the jokes are sexist/racist/homophobic, they're also not funny and clichéd. In the first episode the Father said something along the lines of ' I wish women didn't go out and get jobs and have the same rights as men blah blah blah' That really helps attitudes huh? Then he was making fun of his son saying he was weird. What parent says their kid is weird? So overall this show is boring, unoriginal, offencive, clichéd and most of all NOT FUNNY. Yeah American Dad's offencive. But it does also make you laugh and is obviously taking the micky. Thats the difference.”

Methodology

1. Data preprocessing

   The first step in the solution is preprocessing the text data to remove anything that will make the classification harder and will confuse the classifier, this includes stop-words removal , HTML tags removal and punctuation removal and lemmatization (getting roots of words).

   i-HTML tags removal:

   It can be clearly noticed from dataset exploration that the reviews contains HTML tags which are not be useful in the classification and may confuse the model and be interpreted as features. so the first preprocessing step was removing those tags by using Beautifulsoup library.

   ii-Punctuation and stop-words removal

   An assumption is made that all punctuations like commas, apostrophes,exclamation marks ...etc and all stop-words like am , is , are,the….etc are not good features to keep and won’t help in classification so they were removed using NLTK text processing library.

iii-Lemmatization

Lemmatization is the process of getting the linguistic root of a word,it is better than stemming because for close words produces same output which helps in calculating term frequency in text.

Lemmatization to be done efficiently needs the pos-tag of words which is its position in the sentence and usage (Noun,Verb,Adjective,Adverb).

Also, inside lemmatization function some unhelpful words were removed because of the hypothesis that they are present in all reviews disregarding the sentiment like (movies,films,will,can)

Example input : telling, touched

output: tell ,touch

An assumption is made that returning words to their roots will help better in feature extraction step because it uses word frequencies in documents and this will help similar words from the same root be interpreted as one one and consequently have more frequency value.

2. Feature Extraction

The second step would be to modify the shape of data for the classifier and this includes transforming data into vectors using one of word embedding techniques (TF-IDF , word2vec , Doc2vec).

TF-IDF (Term Frequency - Inverse Document frequency) is a numerical feature extraction method for text data used to get more information about the text using Bag of words representation. ngram_range that is given as a parameter is the number of adjacent word associations made and I chose them to be pairs.

TF-IDF method was chosen because Bag of Words methods according to the paper gave the highest results and other newer feature extraction methods like word2vec and doc2vec were not tried.

3. Supervised learning

   The third step to be able to classify reviews is training a supervised classifier such as SVM,Decision Trees,Logistic Regression or one of ensemble methods on a subset of data (the training set) and testing performance of learning on another subset (the testing set) to ensure that the model generalizes well and be able to judge the performance.

   Using Logistic regression and Naive Bayes supervised learning classifiers, fitting them on input training vectors then outputing the classification accuracy as a percentage.

   I chose Naive Bayes because it is famous for its good results with text data and Logistic Regression because it was the only classifier allowed in the 2016 kaggle challenge and it produced good results in binary classification.

   The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification).

Algorithms and techniques:

Naive Bayes is a probabilistic classifier that uses the probability of appearance of features in certain classes to predict the correct class of a given input but it makes an assumption of having independence between features.It is a conditional probability model given a problem instance to be classified, represented by a vector X=(x1,x2...xn) representing some n features (independent variables), it assigns to this instance probabilities P(C_k/ x1,x2...xn) for each of K possible outcomes or classes C_k.

We Classify and give a label to the review according to the class that has the highest conditional probability given the features as conditions.

Logistic Regression is an algorithm mostly used for binary classification by fitting a the best curve possible through tuning linear equation coefficients to make a decision boundary between classes through searching for minimal mean squared loss.

Logistic regression uses an equation as the representation, input values (x) are combined linearly using weights or coefficient values (b1):

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)).

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The output y is a probability of a certain class and it is transformed to a label (one of two binary labels).

(resource: https://machinelearningmastery.com/logistic-regression-for-machine-learning/)

A good thing about Logistic Regression beside the fact that it runs in little time in this task is the fact that it supports online learning (there are implementations of it that support it), this means that we can add more training examples if they become available in the future without needing to rerun the algorithm on the whole dataset again.

(resource: “A Family of Online Boosting Algorithms” by Babenco et al.)

Gridsearch is a hyperparameter optimization algorithm that tries all input hyperparameter values given on some portion of data to find best hyperparameters and best model to operate on the data.

Process Complications

In the process of coding the only thing I struggled with was in the tuning step where I used train_test_split() function of scikit learn to get 10% of data to use for tuning but the tuning wouldn't improve accuracy based on those splits and hyperparameters did not change but when I hand split the 10% I wanted the tuning worked and accuracy improved and this is probably because I chose some biased unshuffled data but they seemed to work for tuning, so it was not fully automated but I also used trial and error.

Results

I- Logistic Regression

Before tuning

The obtained classification accuracy with the default hyperparameter values (C=1) was 86.868%.

After tuning

GridsearchCV was used to tune the C hyperparameter of the Logistic regression Classifier on 10% of training set, where C parameter is the inverse of the regularization strength.

Obtained accuracy: 87.928 % at C=50

II-Naive Bayes

Before tuning

The obtained classification accuracy with the default hyperparameter values (alpha=1) was 84.676 %.

After tuning

GridsearchCV was used again to tune the alpha hyperparameter of the Multinomial Naive Bayes Classifier on 10% of training set, where alpha parameter is the smoothing factor.

Obtained accuracy: 84.6319% at alpha=0.5 .

K-Folds Cross Validation:

Besides hyperparameter tuning using Grid Search, K-Folds Cross Validation is now used to validate the robustness of the model by making several train-validation splits within data and train the model on them and test on different validation sets, to detect overfitting and underfitting signs.

The number of folds was chosen to be 5, so that 20k samples are used for training and 5k are used for validation and the mean accuracy obtained was 88.92% which is very good and indicates a good fitting curve.

Scores on 5 different folds: [88.78 88.24 88.9 89.44 89.24]

As seen the variance from the obtained accuracy on test data is less than 2% which great indication and proof of the robustness of the model.

Model Evaluation and Validation

The tuned C parameter of logistic regression is the inverse of regularization strength (higher values of C correspond to less regularization) which means on increasing the value of C we allow the equation to get to higher degrees in order to fit the problem better which results in more complexity. So by tuning the best value for C found that gives best accuracy was 50 as compare to the default value 1 which means we need to decrease regularization and allow more degrees of freedom to make better classification. But in comparing results we find that the accuracy was not improved by much (about 1.2%) so this improvement can be removed if we wanted less complexity.

The alpha parameter for naive bayes is additive (Laplace/Lidstone) smoothing parameter: 0 for no smoothing and 1 for full smoothing. Smoothening aims to reduce the effect of rare words in classification, for example: if a word "hot" is found only in one negative sentiment review , we want to reduce the effect of it in classification of all reviews containing this word as negative just because of it. The accuracy values show that the default smoothening value (alpha=1) gave better results by a little amount.

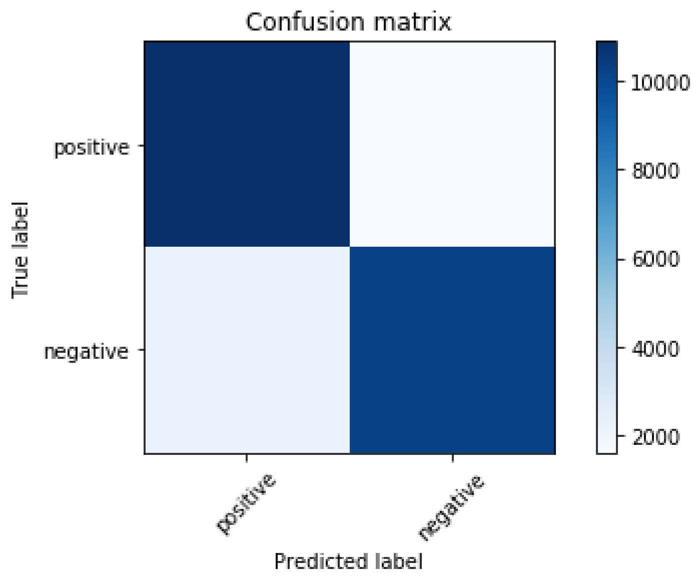
Justification

The best accuracy obtained was that of logistic regression equivalent to 87.928 %, which is much better than the random chance benchmark (50%) and very close to the other stated benchmark by Maas et al. (88.89%) only off by 1%. So this solution succeeds to solve the problem to a good extent and can classify most text sentiments of movie reviews especially that it had been trained on 25k different reviews.

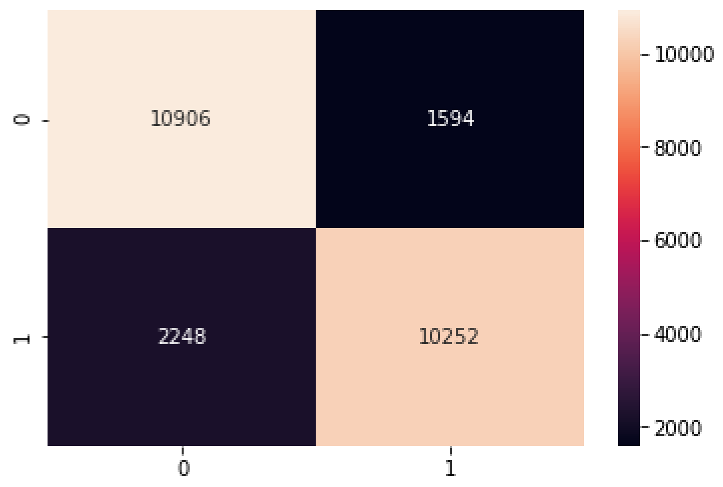
Conclusion

Free Form Visualization (Confusion Matrix)

By plotting the confusion matrix we see that predicting positive sentiment is the best performance of the model is in predicting True-positive labels and there is some sort of diagonal symmetry within the matrix, so the model is balanced and does not tend towards one type of misclassification over others by a big value.



```
[[10906  1594]
 [ 2248 10252]]
```



Reflection

The project summary is that in order to classify the sentiment of reviews I needed to preprocess the text first to remove any words that don't help in classification and then extract numerical features based on word frequency in documents to give weights to words and finally use supervised learning to process those features and make decision boundaries to classify any test document given its numerical features.

What I found challenging about the project was understanding word embeddings and extraction of features from text but it was a thing worth learning. Another challenging thing was all the documentation required and explanations of each step which is so time consuming but it is necessary to produce any readable output that is valuable to others.

The boring things was needing to wait a long time until lemmatization is finished as it is a very time consuming task.

Also as mentioned in the “Methodology” section the hyperparameter tuning was not straight forward.

Improvement

I think what can be improved is trying another word embedding models like word2vec and doc2vec to extract features in other ways and maybe use other classification methods like deep learning ones (CNNs or RNNs or Recurrent Neural Networks).

Also by looking at the samples , there are spelling mistakes so a number of words will be interpreted differently and consequently their frequencies in text will not be accurate, so adding a spell checker and corrector would be good.

Credits

InProceedings{maas-EtAl:2011:ACL-HLT2011, author = {Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher},

title = {Learning Word Vectors for Sentiment Analysis},

booktitle = {Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies},

month = {June},

year = {2011},

address = {Portland, Oregon, USA},

publisher = {Association for Computational Linguistics},

pages = {142--150},

url = {http://www.aclweb.org/anthology/P11-1015} }

References

Potts, Christopher. 2011. On the negativity of negation. In Nan Li and David Lutz, eds., Proceedings of Semantics and Linguistic Theory 20, 636-659.