Group Members: Eric Eli, Tyler Snow

Computer Science 4750 Project

November 30, 2018

Implementing Naive Bayes, Support Vector Machines for Predicting Sentiment Analysis

1. Abstract

We have chosen to implement two different sentiment analysis algorithms as we believe this to be one of the most interesting fields of NLP since it is applicable to so much of human life and our written interactions. More specifically we chose to implement Naive Bayes classification and SVM as it is a great choice for extracting human emotions, opinions, and attitudes from text. (Liu, 2012, p. 24) While there has been a large amount of research in this field, there is a lot of shortcoming to these techniques (Liu, 2012, p. 36) and we wanted to explore these more in depth. Sentiment analysis can also be applied to many real-world applications in any profession, Since the expressed opinions of people have an impact on society. In the business world these translate from computer science to management science as opinions about products often are reflected in profits. (Liu, 2012, p.133)

2. Sentiment Analysis Algorithms

2.1 Naive Bayes Classifier

The Naive Bayes Classifier uses Bayes Theorem (Medhat Et al.,2014, p. 1099) and expands it to n dimensions where n is the number of features (words) to calculate the probability of different parts of the data being associated with a certain class. In this case there are two classes, positive sentiment and negative sentiment.

Bayes' Rule is:

$$P(A \mid B) \;=\; \frac{P(A) * P(B|A)}{P(B)}$$

Where:

$P(A \mid B)$is the probability that A occurs given that B is true

$P(B \mid A)$is the probability that B occurs given that A is true

$P(A)$ and $P(B)$are the probabilities of A and B occurring independently

As stated above, the Naive Bayes Classifier expands this rule to n dimensions and uses it to classify the data

$$P(A \mid x_1,\, x_2, \ldots, x_n) \;=\; \frac{P(A) \;* \prod_{i\,=\,1}^{n} \quad P(x_i \mid A)}{\prod_{i=1}^{n} \quad P(x_i)}$$

Where,

$P(A \mid x_1, x_2, \ldots, x_n)$is the probability that A occurs given $x_1, x_2, \ldots, x_n$is true

$P(x_i \mid A)$is the probability that $x_i$occurs given A is true

$P(A)$ and $P(x_i)$are the probabilities of A and B occurring independently

The name Naive Bayes comes from the naive assumption it makes about the independence of the features (x_i). It is this assumption that lets us use $P(x_i \mid A)$instead of the $P(x_1 \mid x_2, x_3, \ldots x_n, A)$

Since the denominator of this equation will be constant for future calculations you can discard it from the equation

2.2 Support Vector Machines

A Support vector machine takes a set of n-dimensional vectors and uses hyperplanes (i.e. an (n-1)-dimensional surface) to linearly separate the data into distinct classes. Since text data is ideally suited for SVM classification due to the sparse nature of text, it tends to be easily sortable into linearly separable categories. (Medhat Et al.,2014, p. 1100) Support Vector Machines are very universal learners and the most basic SVMs learn linear threshold functions (Joachims, 1998, p. 2) which was ideal for our problem when dealing with positive and negative reviews.

The general approach of SVMs is to find a hyperplane represented by a vector $\underline{\omega}$.This vector separates the document vectors from one another but also tries to maximize the margin of separation between them. This corresponds to a constrained optimization problem, by letting $c_j \in \{1, -1\}$ (that is corresponding positive and negative) be the correct class of the document d_j which gives the formula:

$$\underline{\omega} = \sum_j \alpha_j c_j \underline{d_j}, \alpha_j \geq 0$$

Where the α_j's can be obtained by solving a dual optimization problem. (Pang Et al., 2002, p. 4) Using this equation we can generate the support vectors and so the classification of test instances simply determines which side of $\underline{\omega}$'s hyperplane they fall on.

3. Comparisons between algorithms and their success rates

The Naive Bayes algorithm was implemented from scratch. However, the Support Vector Machine algorithm used the sklearn package. The support vector machine algorithm quickly became very complicated and difficult to generate a working model. Since we were unable to get it working properly, we chose to import the sklearn package to utilize theirs. Once we were able to get both models trained and began testing the sklearning's SVM model was performing much more effectively than our Naive Bayes. This is likely due to a large number of optimizations and data preprocessing that we were not able to implement in our Naive Bayes due to a lack of resources and time.

After training both of our algorithms on a dataset of 25000 IMDB movie reviews (half positive, half negative) and testing them on three different datasets of different sizes gave the following results.

Data set	**Naive Bayes**	**Support Vector Machine**
IMDB Movie set of 25000 Reviews	46.704%	84.444%
IMDB Movie subset of 650 Reviews	47.121%	84.999%
Stanford Movie Set of 2000 Reviews	45.150%	82.499%

Given these results we can conclude that the SVM implemented using Sklearn packages is within the accepted accuracy range of 70-90%(Jain Et al.,2016,p. 4) but our Naive Bayes was not in this range

4.Software Development Procedure

4.1 Finding Data

For most of our testing we used a very large set of over 50000 IMDb reviews. (Maas Et al., 2011) These reviews were split into half positive and negative and those were subsequently split to half testing and half training. We also used a dataset of 2000 IMDb reviews that were also half positive and half negative. (Pang Et al., 2004)

4.2 Preprocessing data

The datasets we used to train the algorithms consisted of movie reviews from IMDB, these data sets were not preprocessed and still included the html code to break the lines. Before using the data to train and test the algorithms it was preprocessed and tokenized. To do this all the punctuation and some unwanted text features were removed. Some of these features are break tags left in the reviews when they were collected and singular letters leftover from the removal of punctuation in compound words and contractions. For example, the singular s left over from removing " ' " from it's. Then the dataset was split into the individual words by white space and put into a list.

In the case of SVM this tokenized list was then vectorized to an n-dimensional vector where n is determined by the number of unique words. After being vectorized the SVM was able to both train and test on the data.

4.3 Implementing algorithms

4.3.1 Naive Bayes Classifier

To train the classifier, we pass the path to the dataset to the Naive Bayes class. The files in the dataset were read and tokenized one by one and the counts of each word in positive and negative files is counted. Once we have total counts for all words in the dataset we compute the necessary probabilities.

$P(x_i \mid A)$ is the probability of the word occurring given sentiment A. So if A is the positive sentiment,$P(x_i \mid A)$ is the count of word x_i occurring in a file marked with a positive sentiment divided by the sum of x_i occurring in both positive and negative sentiments. Since these probabilities are so small, their products tend to be very susceptible to underflow. As such, we used the sum of the natural logs of the probabilities to avoid this problem.

4.3.2 SVM Classifier

For our SVM classifier we utilized the SKlearn package to implement it. More particularly we used their Vectorizer function to vectorize the data as well as their SVC.Linear function which uses a linear function to classify the data. We then stored both of these in an Sklearn Pipeline object which stores the classifier function and vector so that they will not have to be fit to each other each time before testing. Once we created this pipeline object we train it on a given set and that trained model is written to a file using the pickle package in python.

To test the model we would read in the trained Pipeline file then use it to test on the dataset that was passed to it. Once the test completes the accuracy is displayed to the user.

4.4 Integrate Software

_____To integrate our algorithms into a user friendly environment we built a UI that allows the user to simply select the datasets they wish to use a push a test or train button. We used the python library Tkinter to build this UI which is simply but robust and functional for the purposes of training and testing our algorithms. When the user is ready to test the trained models the UI will display the results of the both the algorithms and their accuracy with respect to each dataset.

5.Difficulties and Challenges

5.1 Acquiring Data

It was very difficult finding free and somewhat clean data we could use. We contacted ImDb inquiring about accessing their movie review data through some sort of API but received a strongly worded “no” in response. We also looked at accessing review data from Rotten Tomatoes and saw that other people looking for the same data being quoted a rather large sum of money for very limited access to this data.

Most of the free data we were able would require quite a lot of cleaning and writing a custom parser for each different set.

5.2 Tuning Naive Bayes

While the implementation of the Naive Bayes classifier was straight forward, tuning performance was a challenge. Without the removal of some stop words, precision was about 30%. This is likely due to the probabilities of stop words overpowering the meaningful features and the abs. Once we did implement stop words, our precision increased 17%.

6. Conclusion

Based on our limited datasets we found that out of the box, Naive Bayes was demonstrated to not be very accurate when used with large sets. Whereas SVM was shown to be effective under the same constraints. With some optimizations such as removing more stop words and a combination of bigrams and unigrams to preserve some of the ordering lost from the bag of words implementation, the accuracy of Naive Bayes can be improved.

7. Program User Instructions:

7.1 Run Program

- 7.1.1 To run the application the user must first navigate to the /src folder in the terminal.
- 7.1.2 The user must then run the command "python ui.py" which will display the UI for our program

7.2 Training

- 7.2.1 To train models select the train tab at the top of the application.
- 7.2.2 From there select which model you wish to train (Naive Bayes or Support Vector Machines) and the respective data set.
- 7.2.3 NOTE: Our program will automatically overwrite the most recent training model for each algorithm.

7.3 Testing

- 7.3.1 Using the test tab at the top of the application the user must select what set(s) they wish to test the models on.
- 7.3.2 Then select the test button and wait for the tests to complete (May take a minute depending on size of testing sets).
- 7.3.3 Once completed the results of each algorithms accuracy relative to each test set will be displayed in the grid at the bottom of the test tab.

7.4 Adding Custom Sets

- 7.4.1 If the user wishes to use their own custom data they should add them to (PathToProject/Project/Datasets/custom(1,2 or 3).
- 7.4.2 Sets must then be placed in the test or training folders depending on what the user wishes to use them for.
- 7.4.3 Sets must be given in the form of text files that are stored in folders named '/pos' or '/neg' depending on their sentiment.

References:

[1] Anuja P Jain, Padma Dandannavar, “Application of Machine Learning Techniques to Sentiment Analysis”, 2nd International Conference on Applied and Theoretical Computing and Communication Technology(iCATccT), 2016

[2] Walaa Medhat, Ahmed Hassan, Hoda Korashy, “Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal” (p.1093-1113) ,2014

[3] Bing Liu, “Sentiment Analysis and Opinion Mining”(p.219-248). Morgan & Claypool Publishers, 2012

[4] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques”, Proceedings of EMNLP(p.79-86),2002

[5] Thorsten Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, In Proceedings of the European Conference on Machine Learning (ECML), Chemnitz, Germany, 1998

Dataset References:
[6] Andrew Maas, Raymond Daly, Peter Pham, Dan Huang, Andrew Ng, Christopher Potts, “Learning Word Vectors for Sentiment Analysis”(p. 142-150) (IMDB Movie Dataset), Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, June 2011, http://www.aclweb.org/anthology/P11-1015

[7] Bo Pang, Lillian Lee, “Movie Review Data”(polarity_dataset_v2.0), June 2004, http://www.cs.cornell.edu/people/pabo/movie-review-data/

Python Packages used:
Tkinter- Used for developing UI
Scikit-learn - Used for implementing SVM