Sentiment Analysis on IMDB movie review dataset using SVM

Sameer Koleshwar (202SP010), Mayank Rajpurohit (202SP015)
Department of Electronics and Communication Engineering,
National Institute of Technology Karnataka, Surathkal

May 2021

Abstract – With the growing accessibility of social networking, Sentiment Analysis has become one of the most prominent research domain in natural language processing. Every day, millions of people share their thoughts and ideas by posting in social media or writing online reviews. This massive participation, on one hand, makes these media opinion rich. However, on the other hand, it poses some challenges in identifying the dominant opinion. In this work, we try to classify movie reviews of the famous IMDB's movie review data. The classifier used is Support Vector Machine with TF-IDF weighting as improvement of accuracy before classification.We tried to implement a Support vector classifier from scratch and trained it to compare results. Four different evaluation metrics: recall, precision, accuracy and F1 score are used for evaluating the results of our system.
Keywords: Sentiment Analysis, Support Vector Classifier, Supervised Learning, TF-IDF.

1 Introduction

The development of social media is very rapidly coupled with the increasing number of people who express opinions, submit criticism and poured his thoughts in social media. Among various types of social opinion rich resources, Twitter and Movie review blogs are very popular to share views on different issues or on specific movies. For observing and exploring people's thoughts, Sentiment Analysis has become most prominent research area as it aims to verdict hidden patterns in a large number of reviews, tweets, etc. The initial step of our Sentiment Analysis process incorporates data preparation [1].

We are using famous IMDB's movie review data set which contains 50,000 reviews in total out of which 25,000 are positive and 25,000 are negative. Selection of the suitable pre-processing methods come out to be a great step in improving the classification accuracy. Stop Words removal, remove URLS, @, # and other special symbols, remove numbers, Stemming are the pre-processing steps that we applied on our data. Furthermore, we emphasized on syntactical features because our main focus was on the content of reviews.

We followed the n-gram model [1] for feature extraction & we worked with unigram, bigram, trigram and combination of unigram, bigram & trigram with two different weighting schemes viz. term frequency-inverse document frequency (tf-idf) and binary(bag of words). The ready to fed data then fed to a Support Vector Classifier which is constructed from scratch and training of support vector machine classifier is done with the help of using labelled data. After this classifier model is build which is able to classify the test data.

2 Background

2.1 Data and its pre-processing

The dataset is IMDB's movie review dataset labelled with two categories, 25,000 positive reviews and 25,000 negative reviews i.e 50,000 in total. Data cleaning is the first step performed. Unwanted data such as hashtags, html syntax, and emoticon will be removed. After that, the pre-processing step such as converting to Lower Case, Tokenization, and Stemming is taking place.

Lower case or Case Folding is a step to change every uppercase letter in the data to lowercase. Tokenization is converting sentences in a document into a data set and eliminate delimiter such as periods (.), Commas (,), spaces and numeric characters, to get tokens or term forms. Another name for Tokenization is Parsing, a process that is carried out by someone to make a sentence more meaningful or to be in a way by breaking the sentence into words or phrases. Tokenizing can be done in one of the libraries in Python, the NLTK library. The stemming process is the process of cutting or removing affixes from a word. Those words then used as basic words in the stemming process carried out in the NLTK library. After going through the series of processes above, data in the form of term/feature list that will be grouped by the label will be produced [2] [3]

2.2 Transformation

At the transformation stage a feature selection process was carried out. Selection feature was a process changing value from categorical to numerical data. The stages of transformation were carried out using TF-IDF. The term frequencies document frequency inverse (TF-IDF) is a word weighting process where the word will be extracted into the form of a value. This process is used to assess the weight of the term relevance of a document to all documents. Term frequency is a measure of how often a term appears in a document[2] [3].

$$TF = \frac{Frequency\ of\ the\ word\ in\ sentence}{Total\ number\ of\ words\ in\ the\ sentence} \tag{1}$$

IDF is the number of certain terms in the whole document.

$$IDF = \log\left(\frac{Total\ number\ of\ sentences}{Number\ of\ sentences\ containing\ that\ word}\right) \tag{2}$$

The binary data produced by the process then become parameters so it can be processed in the classification process.The purpose of inverse document frequency is to give weight to the terms found in many documents. The TF-IDF size is a multiplication term between TF and IDF, while the TF-IDF results for one document in the corpus add up the total weight of terms in a document[2].

2.3 Classification

Support Vector Machine (SVM) used as a classification's method to get a value or pattern that results from training on SVM classification. The pattern will be saved into a model that aims to predict labelling in the testing phase. SVM is a supervised machine learning algorithm that has been used for both classification and regression problems. SVM classifies by determining a hyperplane to classify the data distributed in the n-dimensional space. The classification is done based on the mathematical functions called kernels and these kernels are used to determine a hyperplane. Two different classes exist on the opposite sides of the hyperplane and thus this

plane could be considered a decision boundary which could help in simple classification of the data points available. The equation of the hyperplane is as follows:

$$w^T x - b = 0 \tag{3}$$

Where w = Weight vector, x = Input vector and b = Bias[4].

3 Support Vector Machine

3.1 Background

Let $x \in \mathbb{R}^n$ be a training data point, and $y \in \{1, -1\}$ is its label. Suppose you want to find a hyperplane which separates all the points with -1 labels from those with +1 labels. The hypothesis to separate the points is a hyperplane, i.e. a linear subspace that splits all of $\mathbb{R}^n$ into two halves. The data that represents this hyperplane is a single vector w, the normal to the hyperplane, so that the hyperplane is defined by the solutions to the equation

$$\langle w, x \rangle = 0 \tag{4}$$

The key intuitive idea behind the formulation of the SVM problem is that there are many possible separating hyperplanes for a given set of labeled training data. The assumption of the SVM is that a hyperplane which separates the points, but is also as far away from any training point as possible, will generalize best.

$$y_i = \text{sign}(\langle w, x \rangle) \tag{5}$$

w encodes the following rule for deciding if a new point has a positive or negative label. You'll notice that this formula only works for the normals w of hyperplanes that pass through the origin, and generally we want to work with data that can be shifted elsewhere. We can resolve this by either adding a fixed term $b \in \mathbb{R}$—often called a bias because statisticians came up with it—so that the shifted hyperplane is the set of solutions to $\langle x, w \rangle + b = 0$. The shifted decision rule is:

$$y_i = \text{sign}(\langle w, x \rangle + b) \tag{6}$$

Now the hypothesis is the pair of vector-and-scalar w, b.

3.2 Crammer-Singer multiclass SVM formulation

SVMs classify an input vector $x \in R_d$ into one of k classes using the following simple rule:

$$y_i = \underset{m \in [k]}{\text{argmax}}\, W_m^T x \tag{7}$$

Each vector $W_m \in R_d$ can be thought as a prototype representing the mth class and the inner product $W_m^T x$ as the score of the $m^t h$ class with respect to x. Therefore, Eq. r(1) chooses the class with highest score. Given n training instances $x_i \in R_d$ and their associated labels $y_i \in [k]$, Our optimization problem is the following (including the bias again):

$$\begin{aligned} &\min_w \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad &(\langle x_i, w \rangle + b) \cdot y_i \geq 1 \quad \text{for every } i = 1, \ldots, m \end{aligned} \tag{8}$$

This is much simpler to analyze. The constraints are all linear inequalities (which, because of linear programming, we know are tractable to optimize). The objective to minimize, however, is a convex quadratic function of the input variables—a sum of squares of the inputs.

The Crammer-Singer multiclass SVM formulation [3] estimates $w_1, \ldots, w_k$ by solving the following optimization problem:

$$\underset{w_1,\ldots,w_k}{\text{minimize}} \frac{1}{2}\sum_{m=1}^{k}\left\|w_m\right\|^2 + C\sum_{i=1}^{n}[1 + \max_{m\neq y_i} w_m^T x_i - w_{y_i}^T x_i] \tag{9}$$

where C > 0 is a regularization parameter and $[u]_+ = 0$ if $u < 0$ and u otherwise. Intuitively, 9 Eq. (9) means that, for each training instance, we suffer no loss if the score of the correct class is larger than the score of the "closest" class by at least 1. In the remainder of this paper, we assume that $\left\|x_i\right\| > 0$, since any $\left\|x_i\right\|$ with$\left\|x_i\right\| = 0$ (which can only happen if $\left\|x_i\right\| = 0$) does not affect the solution of Eq. (9)[4].

3.3 Algorithm

Algorithm 1 Algorithm of Crammer-Singer Formulation

INPUT: X as train data, Y as train labels
X = hstack([X, ones((len(X), 1))])
N, d = X.shape
k = no. of classes
Initialize W matrix of size (k, d), penalty constant C = 1.0 , i=0
while $i < iterations$ **do**
    $grad = gradient(W, X, Y, C)$
    $W = W - \alpha * grad$
    $i = i + 1$
end

The Crammer-Singer Algorithm gives us the alternate optimization problem to solve for the Support Vector Classifier. The algorithm to solve that optimization problem is explained above. First we initialise W matrix. Then a column of ones is stacked horizontally to the input data , in order to accommodate bias b within the parameter W itself. Then for n no. of iterations we perform gradient descent steps. Calculating the gradient w.r.t W, gradient of regularization term will be W. gradient of sum term is: x_i for positive case and $-x_i$ for negative case and 0, otherwise. Hence gradient will be $w + x_i.x_i$ for positive case and $w - x_i.x_i$ for negative case. Then the next step will be updating W as $W = W - \alpha * gradient$. At the end we get W matrix of trained weights which can be used to predict labels for new input data.

3.4 Performance Matrices

3.4.1 Accuracy

Classification accuracy is a metric that summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions. It is easy to calculate and intuitive to understand, making it the most common metric used for evaluating classifier models.

$$Accuracy = \frac{Correct\ predictions}{Total\ Predictions} * 100$$

3.4.2 Precision Recall and F1 Score

F1 Score is the harmonic mean of Precision and Recall.Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.Precision is a good measure to determine, when the costs of False Positive is high.

$$Precision = \frac{True\ Positive}{True + False\ Positive}$$

Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Hence, F1 score is

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1 Score is needed when you want to seek a balance between Precision and Recall. F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall and there is an uneven class distribution (large number of Actual Negatives).

4 Results

Results in the form of before mentioned Performance metrics has been summarized below. Fig.1 gives accuracy for both the models trained using sklearn and from scratch for 1-gram, 2-gram and 3-gram modelling techniques. And the Table.1 summarizes Accuracy, Precision , Recall and F1 scores for every model trained.

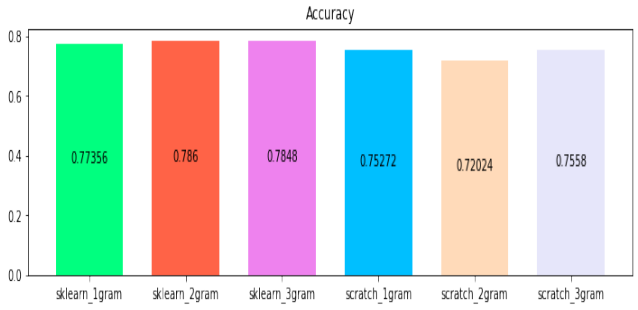

Figure 1: Accuracy

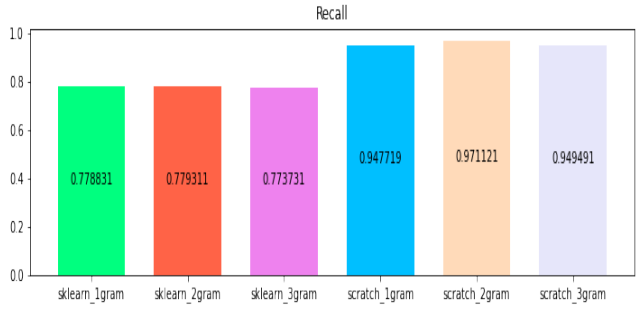

Figure 2: Recall

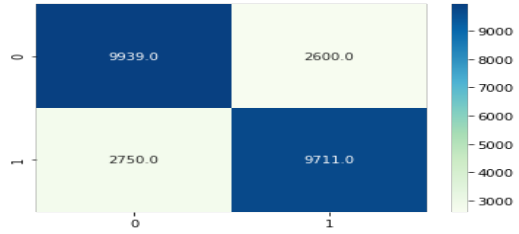

Figure 3: Confusion Matrix for 2-gram TF-IDF Model build using sklearn

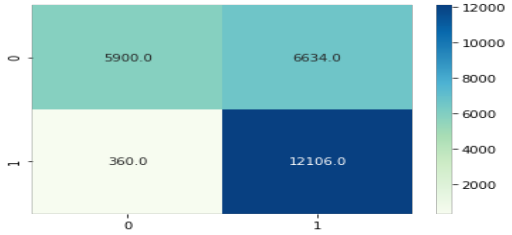

Figure 4: Confusion Matrix for 2-gram TF-IDF Model build from scratch

Models	Precision	Recall	F1-score	Accuracy
1-gram using sklearn	0.772	0.778	0.775	0.773
2-gram using sklearn	0.788	0.779	0.784	**0.786**
3-gram using sklearn	**0.792**	0.773	0.783	0.785
1-gram from scratch	0.681	0.948	0.792	0.753
2-gram from scratch	0.646	**0.971**	0.776	0.720
3-gram from scratch	0.683	0.945	**0.795**	0.756

Table 1: Perfomance metrices for every model

5 Conclusion

Based on the results and discussions presented above, the conclusions are obtained as below:

1. In the context of our work, we present an SVM based classifier from scratch that uses tf-idf and n-gram based features to train model for sentiment classification.

2. We conducted experiments with three different n-gram feature sets. Out of which the 3-gram features show best performance in term of accuracy and F1 score result compared to other feature sets.

3. Accuracy metrics might give us misguiding results for classification tasks. F1 score is a reliable in those situations.

4. Regarding the future work, we are planning to explore some more relevant external knowledge that can serve as a feature set in order to provide further improved performance.

References

[1] Aditi Sharan Sheeba Naz and Nidhi Malik. ""sentiment classification on twitter data using support vector machine". In: *IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (2018).

[2] Casi Setianingsih Rimba Nuzulul Chory Muhammad Nasrun. "Sentiment analysis on user satisfaction level of monile data services using support vector machine (SVM) algorithm". In: *IEEE International Conference on Internet of Things and Intelligence System* (2018).

[3] Andia Enggar Mayasari and Anggit Dwi Hartanto. "User Satisfaction Levels Sentiment Analysis Toward Goods Delivery Service On Twitter Using Support Vector Machine Algorithm (SVM)". In: *International Conference on Information Technology, Information system and Electrical Engineering* (2019).

[4] Yoram Singer Koby Crammer. "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines". In: *Journal of Machine Learning Research* (2001).