Sentiment Analysis

Varun Negandhi

Computer, University of Memphis, Memphis, TN, USA, vsngndhi@memphis.edu

1. ABSTRACT

The internet has an increasing collection of data and one of the major contribution to this data is the review text concerning the movie or products bought online. It is a very easy activity for humans to help identify whether a written text indicates a positive or a negative sentiment. This project aims at identifying the user sentiment from the user written sentences through a supervised learning approach. There are 2 solutions proposed for this project – Machine Learning with SVM and Deep Learning with keras.

1. **INTRODUCTION:**

The dataset for this project is the IMDB movie review dataset. This dataset contains 50000 movie reviews and 2 columns – one with review text and the other column is the sentiment (positive or negative) [1]. This was used as a training dataset in both approaches. The test dataset consists of 25000 movie reviews with same 2 columns as present in the training data.

1. **APPROACHES:**
   1. **Machine Learning - SVM** 
      1. **Preprocessing:**

The dataset containing movie review texts contains html texts like <br> which are not required in the text analysis. Hence the first pre-processing part is removing these tags from the text. Now the data remaining is a clean review text data. But text data cannot be used directly for analysis. Hence, we need numerical vector representation of the text data. All the words in the text column of dataset will be treated as vocabulary. It is then converted to numerical vector using tf-idf vectorizer [2]. The data is now ready to be used as input for the algorithm.

* + 1. **SVM Algorithm:**

The preprocessed data was then applied to SVM algorithm using sklearn machine learning package in python [3]. For obtaining the correct set of parameters, the values tested were the cost factor and the type of kernel in SVM. The types of kernels tested were linear, poly, rbf and sigmoid. The best type of kernel was chosen from the best performance of the algorithm parameters. For cost factor in SVM, it is a value which finds the best balance between margin width and the misclassification error. This value can be obtained only through experimentation.

**3.1.3 Evaluation:**

The evaluation metric for the performance of this project is F-score. The tables below give the performance of SVM with different combination of SVM parameters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linear | Poly | Rbf | Sigmoid |
| C = 0.1 | 0.878 | 0.329 | 0.329 | 0.329 |
| C = 1 | 0.879 | 0.329 | 0.329 | 0.329 |
| C = 10 | 0.866 | 0.329 | 0.685 | 0.346 |
| C= 100 | 0.857 | 0.329 | 0.867 | 0.847 |

**Fig 1. F-Score values on train data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linear | Poly | Rbf | Sigmoid |
| C = 0.1 | 0.884 | 0.333 | 0.333 | 0.333 |
| C = 1 | 0.905 | 0.333 | 0.333 | 0.333 |
| C = 10 | 0.917 | 0.333 | 0.685 | 0.346 |
| C= 100 | 0.922 | 0.333 | 0.866 | 0.846 |

**Fig 2. F-Score values on test data**

**3.1.4 Summary:**

The overall performance of SVM on this dataset is good. As the cost factor keeps increasing, the running time of the algorithm keeps on increasing. From the numbers in table above, the linear kernel with cost facor 100 seems to give a very high value for F-score. However, its performance on train and test data varies. So it is appropriate to use Rbf or Sigmoid kernels with cost factor 100 since its performance is closer for train and test data.

**3.2 Deep Learning - Keras**

**3.2.1 Preprocessing:**

The training dataset remains the same as used in the previous approach. The first step in preprocessing involves removing the html tags from the review texts. The next step in this approach is to represent each word in the text as a number. Here, we use a tokenizer [4] wherein all the words in the text are the vocabulary and each word are represented by unique number using a tokenizer. This approach uses deep neural network which takes as input integer values. Also, the target label which is the sentiment here is mapped into integer where positive is 1 and negative is 0. It is required that all the reviews in the dataset are of same length. Hence a constant length of 500 words is chosen for the review text. The long texts are truncated, and the shorter ones are padded using sequence in python.

**3.2.2 Deep Neural Network:**

Deep learning toolkit keras is used in this project to build a deep neural network. The model used for neural network is sequential layer so that experimenting with the number of hidden layers and the number of neurons in each layer becomes easy. The first layer in the architecture is an embedding layer which has parameters of vocabulary size which the number of unique words in the text and the input length which is 500 for the review length. This layer converts each word into a vector of different sizes which are then used for prediction. The number of hidden layer is added after this first layer to experiment with the performance. The output layer consists of 2 neurons for negative and positive sentiments. The output for each review will thus be a vector of shape 1 row and 2 columns which is assumed to be percentage of negative and positive sentiment in a text. the maximum of this value is the predicted label for a given text. For this model, the ‘adam’ [5] optimizer was used to decide on a suitable learning rate. The loss function used was ‘spare\_categorical\_crossentropy’. This model has a high chance of overfitting on the training data and hence the number of epoch chosen is 2 to train the model.

**3.2.3 Evaluation:**

Like the previous approach, the performance metric is F-Score. The values were tested with embedding layer, embedding plus 1 hidden layer and embedding plus 2 hidden layers. Using only the embedding layer gave maximum F-score when compared to other 2 approaches.

A screenshot of a cell phone

Description automatically generated

**Fig 3. F-Score values with embedding layer for train and test data**

From the above figure, the correct set of parameters are embedding layer vector size 30.

**3.2.4 Summary:**

This approach shows greater promise than the SVM with F-score of 94.82 percent. On adding more hidden layers to the embedding layer, the performance did not seem to improve much. Hence only the embedding layer with vector size 30 was considered in this model.

1. **Conclusion:**

The project was overall successful with both the approaches giving good F-scores. The majority of effort needed in this project was with the data pre-processing in both the methods. For SVM approach, as the cost factor was increased, the computation time taken by the algorithm kept increasing. The computation time with deep learning approach was less. The performance in terms of F-score was also better with deep learning approach [7]. An attempt was also made to propose a solution using LSTM [6] but the computation time was too long. So future work can include optimizing LSTM’s computation time and it may yield better performance than the solutions proposed in this report.

5. Reference:

[1]<https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

[2] <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

[3] <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

[4] <https://keras.io/preprocessing/text/>

[5] <https://keras.io/optimizers/>

[6]<https://medium.com/@panghalarsh/sentiment-analysis-in-python-using-keras-glove-twitter-word-embeddings-and-deep-rnn-on-a-combined-580646cb900a>

[7] <https://github.com/vsngndhi/Sentiment-Analysis>