Twitter Sentiment Analysis

DATA MINING AND TEXT MINING- CS583
PROJECT 2
Praveen Chandrasekaran (pchand34)
Sai Krishnan Thiruvarpu Neelakantan (sthiru5)

Abstract:

The purpose of this project is to classify the given twitter tweets in to positive, negative and neutral classes i.e. the project basically analyses the emotion of a particular tweet. To train and classify the given data, We have used different classification techniques such as Multinomial Naive Bayes, Linear SVM(Support Vector Machine), LSTM(Long Short Term Memory), CNN(Convolutional Neural Network) and ensembles of some of the models mentioned above. Before feeding the data into the model, pre-processing of data was done, features like unigram, bigram, trigram, padding, Word2vec etc. were used and finally the data was converted into a format which a machine learning model can understand. Out of all the classification techniques used, prediction taken by majority voting between CNN, LSTM, SVM performed better.

Introduction:

The presence of social networking sites has changed the way how we share our emotions. Millions of messages are appearing daily in different social mediums and for this project we have used twitter messages for analyze the sentiment of the tweets. We have used tweets from 2012 US presidential elections and there are two datasets, one for tweets related to Barack Obama and one for tweets related to Mitt Romney.

The initial data contains empty values in some or all of the columns present for a row which we term as noise and there is also irrelevant data present in the columns which we don't require for training the model. The first step in the pre-processing involves the removal of these noises. Then, removal of irrelevant content from the tweets and was done as the next step in the pre-processing stage. This stage has lots of sub steps and everything is explained in detail in the pre-processing section below. As the final step of the pre-processing, the given data was converted to a matrix representation in order for the machine learning model to understand and features like Bag of Words, TF-IDF, Word2vec were used for this. These features are also explained in detail in the pre-processing section.

Then, different classification techniques were used to predict the classes in the test set. Classification methods section talks in detail about the different methods used in the project. There are several metrics used for computing and comparing the results predicted from the model. Some of the most common metrics used here is Precision, Recall, Accuracy, F1-measure. The evaluation results for each metric and for each model has been added in the below sections. As mentioned in the abstract section, through majority voting, for Obama tweets the accuracy was 62% and for Romney tweets the accuracy was 59%.

Data Preprocessing (Common for both Obama and Romney):

Removed N/A rows from the data:

The null and the N/A values from the field Tweets and Class have been removed. This is because the data is mainly used for training and validation purposes, therefore having these values in the data seemed have no important.

HTML Tags:

The data given had various HTML tags such as <e>, <a> tags. All these tags were removed to clean the data so that only meaningful words are there in the tweets. These tags were removed using the HTML parser using Beautiful Soap.

Hyperlinks, User IDs, Numbers and Special Characters removal:

The tweets is prone to have hyperlinks, User IDs, number and various special characters. To clean the data to all these were removed to from the tweets. It is very hard to derive the sentiment from a hyperlinks, User ID, numbers and special characters. This was done using Regex patterns.

Handled names (Barack Obama and Mitt Romney):

The most common occurring names in the tweets were Barack Obama and Mitt Romney. Since the name cannot express any sentiment, these words were removed from the tweets. This was also performed using Regex patterns.

Tokenization:

These tweets are then converted into a separated array of words for further processing to remove stop words and lemmatization process. The word_tokenize() from the NLTK package was used for tokenizing.

Stop words:

The English stop words such as “the”, “is”, “a”, etc., which are more common in the tweets and don’t express no sentiment in the tweets were removed. The stopwords from the NLTK package were used to remove the stop words from the tweets.

Lemmatization:

Lemmatization takes into consideration the morphological analysis of the words and work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. The WordNetLemmatizer() from the NLTK package was used for the Lemmatization of the words.

Detokenize:

Once the Stop word removal and the Lemmatization is done. The words are then detokened using the join function. From this step the data is processed in a way to fit the models that have been used for classification.

Bag of Words:

The processed tweets are then used to create a Bag of Words. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things vocabulary of known words and measure of the presence of known words. CountVectorizer() function from sklearn and Tokenizer() function from NLTK was used to create Bag of Words based on the classification models.

TF-IDF:

The Term Frequency (TF) – Inverse Document Frequency (IDF) were calculated from the Bag of Words created. TF*IDF is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term. TfidfTransformer() from sklearn was used to calculate TF-IDF scores.

Padding:

Padding was data was done for particular Deep Learning models used for this problem. Since the deep learning models require input to be fed in the same size padding was used. The max size of a tweet is 280 words, the input were padded to max size with 300, filling the remaining spaces in the input with Zeros. pad_sequences() from Keras package was used for padding.

SMOTE (For Romney data):

The given Romney data was a bit skewed at a ratio of 1:3:1.5 for Positives, Negatives and Neutral values respectively. So we used SMOTE (Synthetic Minority Over-sampling Technique) to Over-Sample the negative and neutral class to have a equal ratio of the classes. This enables the model to learn better and predict all the classes equally. SMOTE() from the imblearn package was used.

Glove Weight Vectors:

The Glove Embedding (50 Dimensions) has the pre-trained word weights that can be used to train our deep learning models. The Vocabulary (around 8,000) from our data is compared with the Glove Embedding words (which had around 400,000) and the respective weights were extracted and these weight vectors were used in deep learning models.

Classification methods tried:

Multinomial Naïve- Bayes:

This model was chosen because when it comes to text classification, the Multi-Nomial Naïve-Bayes is the best suited model. The algorithm is specially developed for text classification. MultinomialNB() from sklearn was used to train the model. The naïve bayes model was trained using the TF-IDF vectors created.

Linear Support Vector Machine:

The Linear Support Vector Machine was chosen because the TF-IDF vectors we created has a big dimension. Since, Linear Support Vector Machine algorithm works well with higher dimension data and also is faster this model was used. LinearSVC() from sklearn was used to train the model. The same TF-IDF vectors was used as input for the Linear SVM.

CNN (Convolutional Neural Networks):

Three channels of CNN was build for this problem and the model training was made. Each channel consists of Input, Embedding (50 dimensions along with weight vectors), Conv1D (50 filters, relu activation), Dropout (0.2), Maxpooling and Flatten layers. These three channels were build in such a way that the all the layers take the same input but in the form of Unigram, Bigram and Trigram. Finally the output of all these channels are concatenated and fed in to the Dense output layer with SoftMax Activation function. All these layers are added to the model and the model is then compiled with Categorical Cross Entropy loss, Adam Optimizer and Accuracy as metrics. The model is then trained with the padded input data for 6 epochs with batch size as 64.

LSTM (Long-Short Term Memory):

The LSTM model was used because of its back propagation feedback property. Three layers of LSTM (30, 30 and 20 respectively) along Embedding and Dense output layer was used to build the model. The model was compiled with Categorical Cross Entropy loss, Adam Optimizer and Accuracy as metrics. The model is then trained with padded input data for 5 epochs with batch size as 64.

Ensemble (Voted Classifier):

Another model tried include the Voted Classifier. This takes the output of the Support Vector Machines, CNN and LSTM models and takes the vote of the three models to decide the class. If no majority is achieved from the vote, then the result from the LSTM was finalized as the class.

<u>Model Evaluation:</u>

Obama Dataset:

Multinomial Naive Bayes Text Classifier :
Precision (1,-1,0) → 0.54726368, 0.61111111, 0.6
Recall (1,-1,0) → 0.6626506, 0.66666667, 0.42622951
F1 Score (1,-1,0) →0.59945504, 0.63768116, 0.49840256
Overall Accuracy → 0.5850091
Linear SVM :
Precision (1,-1,0) → 0.58791209, 0.64516129, 0.55865922
Recall (1,-1,0) → 0.64457831, 0.60606061, 0.54644809
F1 Score (1,-1,0) → 0.61494253, 0.625   , 0.55248619
Overall Accuracy → 0.5978062157221207
CNN:
Precision (1,-1,0) → 0.66923, 0.53759, 0.58278146
Recall (1,-1,0) → 0.505813, 0.75261, 0.475675
F1 Score (1,-1,0) → 0.5761, 0.62719, 0.523809
Overall Accuracy → 0.58135
RNN:
Precision (1,-1,0) → 0.745762, 0.52097 , 0.601398
Recall (1,-1,0) → 0.48351 0.82777, 0.464864
F1 Score (1,-1,0) → 0.58666 0.63948, 0.52439
Overall Accuracy → 0.5904
Ensemble (Voted Classifier):
Precision (1,-1,0) → 0.63636364, 0.61607143, 0.61666667
Recall (1,-1,0) → 0.61904762, 0.71875   , 0.53365385
F1 Score (1,-1,0) → 0.62758621, 0.66346154, 0.57216495
Overall Accuracy → 0.62157

<u>Romney Dataset</u>:

Multinomial Naive Bayes Text Classifier :
Precision (1,-1,0) → 0.41714286, 0.64575646, 0.43697479
Recall (1,-1,0) → 0.62931034, 0.63868613, 0.29714286
F1 Score (1,-1,0) → 0.50171821, 0.64220183, 0.3537415
Overall Accuracy → 0.5309734513274337
Linear SVM :
Precision (1,-1,0) → 0.41111111, 0.69230769, 0.48550725
Recall (1,-1,0) → 0.63793103, 0.62408759, 0.38285714
F1 Score (1,-1,0) → 0.5, 0.65642994, 0.42811502
Overall Accuracy → 0.552212389380531
CNN:
Precision (1,-1,0) → 0.4589, 0.67383, 0.48571
Recall (1,-1,0) → 0.6146, 0.6550, 0.4023
F1 Score (1,-1,0) → 0.5254, 0.6643, 0.4401
Overall Accuracy → 0.57168
RNN:
Precision (1,-1,0) → 0.42222, 0.6993 , 0.5277
Recall (1,-1,0) → 0.5643 0.6968, 0.4293
F1 Score (1,-1,0) → 0.4830, 0.6980, 0.4735
Overall Accuracy → 0.5893
Ensemble (Voted Classifier):
Precision (1,-1,0) → 0.6119403 , 0.63220339, 0.4953271
Recall (1,-1,0) → 0.44890511, 0.74749499, 0.46902655
F1 Score (1,-1,0) → 0.51789474, 0.68503214, 0.48181818
Overall Accuracy → 0.5890287769784173

Conclusion:

For this project, we have used different kinds pre-processing and different classification techniques to classify the tweet as positive, negative and neutral. However, there is still a scope for improving the accuracy so that the model can predict better. In case of pre-processing, for our project, we have removed the emoticons from all the tweets in the training and the test data. We can actually use these emojis in classification by giving them a weight so that we can extract more features from the tweets. We could have further improved our classifier by using Weighted Voting Classifier instead of Voting Classifier.

References:

https://machinelearningmastery.com/gentle-introduction-bag-words-model/

https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/

https://www.onely.com/blog/what-is-tf-idf/

https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html

https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/