COMSM0089 Introduction to Data Analytics Coursework

QJ22919/ 2334033

|  |  |
| --- | --- |
| Task 1: Sentiment Classification | |
| 1.1 | |
| **Overview|** | |
| The FiQA dataset consists of JSON files containing Tweets and Headlines along with a label indicating whether the content is mainly positive or negative. Pre processing splits the  JSON into a list of documents and a corresponding list of labels. The document list is processed to remove common words/ punctuation and add negations to negative word pairs (e.g.didn’t like). These document are and then split into tokens (individual words in this case)- this is the ‘Bag of Words’ representation.  Next a word vector is created- all the unique words in the dataset are combined to create a vocabulary for the document. For each document in the corpus a vector is created which is the same length as the vocabulary with a count of the occurrence of the correspondingly located word.  A Multinomial Logistic Regression Classifier is trained in order to learn the weight for each token that best predicts the class labels. For example, ‘Excellent’ is likely to have a strongly positive weight for the positive sentiment class and a strongly negative weight for the negative sentiment class.  The model then predicts the class of documents from the test set by first vectorising the tokens in the document then calculating the weighting for that vector. This weight is then classified using a Sigmoid function.  The performance of the model is assessed by comparing the predictions to the (human) generated labels in the test set. | |
| **Data Pre-Processing** | |
| The data loader iterates through each individual JSON file (Tweet) in the FiQA dataset and adds each’s text and label to a corresponding list.  The list containing the labels for all tweets is then converted from its continuous value ( negative value for negative sentiment and vice versa) into discrete values.  This initial threshold of -0.2 to 0.2 for neutral gave the following distribution of tweets:   |  |  |  |  | | --- | --- | --- | --- | | **Continuous Label** | **Output Label** | **Sentiment** | **Proportion of Dataset** | | Less than -0.2 | 0 | Negative | 28% | | -0.2 & 0.2 | 1 | Neutral | 18% | | More than 0.2 | 2 | Positive | 54% |   From the initial evaluation of the models it was clear that the threshold of -0.2 to 0.2 for neutral gave an model that was heavily biased. Experimentation with the threshold showed neutral between -0.25 & 0.32 produced a model with more intuitive performance   |  |  |  |  | | --- | --- | --- | --- | | **Continuous Label** | **Output Label** | **Sentiment** | **Proportion of Dataset** | | Less than -0..25 | 0 | Negative | 25% | | -0.25 & 0.32 | 1 | Neutral | 32% | | More than 0.32 | 2 | Positive | 42% | | |
|  | |
|  | |
| Negations are appended to words preceded by not, never or words ending “n’t”:  "$AXDX I got bored, and took my proprietary profit exit. I didn't like the option liquidity and lousy spreads."  Becomes:  "$AXDX I got bored, and took my proprietary profit exit. I **not\_like** the option liquidity and lousy spreads." | |
|  | |
|  | |
|  |  |
| The bracketing of the neutral values is arbitrary, a sample of categorised tweets should be examined in order to evaluate that the -0.2 to 0.2 bracket is suitably capturing neutral tweets. | |
|  |  |
| The tweets are split into test, validation and training sets. The test set is used to train the model by providing both the content of the tweet and it’s label .  The validation set is not included when the model is trained so it can be used to give an indication of the models performance when predicting the classification. The predicted values (0/ 1/ 2) for each of the tweets in the Val set is compared to it’s actual label ( which is from the gold data classified by humans).  The hyperparameters of the model can then be tweaked to improve the model’s performance metrics. Only once the model is believed to be optimised is the test split of the data given to the model to predict. This ensures that the model has not been optimised to fit the test set which would give an unrealistic impression of the models performance on other unseen data.  Tokenize from the NLTK package is used in the CountVectoriser to generate word vector for the Test, Validation and Training sets. The Tokenize function first creates a vocabulary of all the tokens ( individual words, punctuation and special features such as email addresses) in all of the tweets within the data split ( or corpus). This is the ‘vectoriser.fit’ stage and is only done against the train dataset as both the validation and test sets are the unseen data for model validation.  Then the CountVectoriser, for each tweet in a dataset, creates a vector the same length as the vocabulary with a count of the times that correspondingly placed word was seen in the individual tweet. This results in a sparse matrix as wide as the vocabulary and with the same number of rows as tweets in the data set. The output form the CountVectoriser is a Bag of Words (BoW) as the the vector for each tweet represents the frequencys of word from the vocabulary being in that tweet but not the order and therefore relation between the words in the tweet.  Normalisation: There are several techniquest that can be applied to standardize the tokens in the vocabulary such as case folding all words to lower case or substituting emoji’s for text. **Only folding to lower case was used here**  The NLTK Tokenize function does?? lemmatize the words to their root form ( reading to read etc.) or stem words ( use regex to drop prefixs & suffixes such as s assuming it means plural).)  I have decide to exclude this step as the data set is small so doesnot require the simpler processing this gives. Also, there is some information loss from stemming the word. | |
|  |  |
| 754 row training set Vocabulary was reduced from 3349 to 3277.  Compressed sparse row format reduced from 10510 to 8351 stored elements. | |
|  |  |
|  |  |
|  |  |
| Naïve Bayes is a generative, probabilistic model which learns the probability of individual tokens being associated with a particular class of document.  Naïve Bayes only considers a single token’s probability so the likelihood of each token is conditionally independent of any other. | |
|  |  |
|  |  |
|  |  |
| No data processing |  |
| Accuracy = 0.6194029850746269  Precision (macro average) = 0.5454545454545454  Recall (macro average) = 0.4776111728166523  F1 score (macro average) = 0.4692675356178211  precision recall f1-score support  0 0.60 0.49 0.54 37  1 0.40 0.08 0.14 24  2 0.64 0.86 0.73 73  accuracy 0.62 134  macro avg 0.55 0.48 0.47 134  weighted avg 0.58 0.62 0.57 134 |  |
| With Data Processing |  |
|  |  |
| Accuracy = 0.6044776119402985  Precision (macro average) = 0.604904904904905  Recall (macro average) = 0.4467018388251265  F1 score (macro average) = 0.44505247376311835  precision recall f1-score support  0 0.61 0.30 0.40 37  1 0.60 0.12 0.21 24  2 0.60 0.92 0.73 73  accuracy 0.60 134  macro avg 0.60 0.45 0.45 134  weighted avg 0.61 0.60 0.54 134 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| End to End Deep Learning Paradigm ( ADA) still applies some preprocessing. |  |
|  |  |
| **Logistic Regression Classifier**  No data processing |  |
| Accuracy = 0.6343283582089553  Precision (macro average) = 0.5518682966209204  Recall (macro average) = 0.5102688304743099  F1 score (macro average) = 0.5163527163527163  precision recall f1-score support  0 0.59 0.46 0.52 37  1 0.38 0.21 0.27 24  2 0.68 0.86 0.76 73  accuracy 0.63 134  macro avg 0.55 0.51 0.52 134  weighted avg 0.60 0.63 0.61 134 |  |
|  |  |
| With Data Processing |  |
|  |  |
| Accuracy = 0.7089552238805971  Precision (macro average) = 0.692927864214993  Recall (macro average) = 0.5594995680612119  F1 score (macro average) = 0.56280577659888  precision recall f1-score support  0 0.79 0.59 0.68 37  1 0.60 0.12 0.21 24  2 0.69 0.96 0.80 73  accuracy 0.71 134  macro avg 0.69 0.56 0.56 134  weighted avg 0.70 0.71 0.66 134 |  |
| Data processing, in particular the negation improved the model for both negative and positive sentiment. The largest error is for neutral statements being classified as positive | |
|  |  |
|  |  |
| With neutral between 0.32 & - 0.25: |  |
| Number of instances: 1111  Number of labels: 1111  Number of negative labels: 283  Number of neutral labels: 359  Number of positive labels: 469 | |
|  |  |
|  |  |
|  |  |
| Accuracy = 0.5671641791044776  Precision (macro average) = 0.5725829725829725  Recall (macro average) = 0.5535835713314293  F1 score (macro average) = 0.5600445316540222  precision recall f1-score support  0 0.61 0.50 0.55 34  1 0.55 0.51 0.53 43  2 0.56 0.65 0.60 57  accuracy 0.57 134  macro avg 0.57 0.55 0.56 134  weighted avg 0.57 0.57 0.57 134 |  |
|  |  |
| By adjusting the threshold scores the split of the data set is adjusted. The actual proportion of positive, neutral and negative tagged tweets is unknown but the confusion matrix above possibly indicates a realistic estimation.  Rather than the model being biased towards positive sentiment as before, each category has more correctly classified than not.  Also, there is a graduation between values which is more intuitive than the earlier model. | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |