COMSM0089 Introduction to Data Analytics Coursework

QJ22919/ 2334033

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task 1: Sentiment Classification | | | | | | | | | |
| * 1. **Implement & Train** | | | | | | | | | |
| **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 documents are split into tokens (individual words in this case)- this is the ‘Bag of Words’ representation.  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 word correspondingly located in the vocabulary.  A Logistic Regression Classifier is trained by finding the optimum weight for each feature in the training word vectors that best predicts the document class ( gold label), this means it is a discriminative model. Cross validation is used to generate possible values of weighting and then gradient descent is used to find the minimum value of the loss function ( a cross entropy calculation).  Logistic Regression is a binary classification method so Multiple Logistic Regression is performed to produce the Positive, Negative and Neutral models.  The dot product of the unseen word vectors and the weighting vector (from the training/ fit stage) is then classified using a Sigmoid function which scales the probability ( an unbounded real number) to a value between 0 and 1.  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 a 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% | | | | | | | | | | |
|  | | | | | | | | | |
| A function *add\_negation* was created which uses REGEX to identify words preceded by *not* or *never* and words which end with *n’t.* The function appends a *not­\_* prefix to these words so that the negative meaning in captured in the word vector. | | | | | | | | | |
|  | | | | | | | | | |
| The tweets are split into test, validation and training sets. The training set is used to train the model by providing both the content of the tweet and its label. A random state value was set for the split so that the evaluation is rep  The validation set is not included when the model is trained so it can be used to give an indication of the model’s performance when predicting the classification. The predicted values (0/ 1/ 2) for each of the tweets in the Val set is compared to its actual label (which is from the gold data classified by humans).  There are hyperparameters in the Logistic Regression Classifier model but the validation set was used to adjust the threshold of the initial conversion from to continuous to discrete labels.  Once the data split was believed to be optimum the test data was given to the model to predict. This ensures that the data has not been optimised to fit the test set which would give an unrealistic impression of the model’s performance on other unseen data. | | | | | | | | | |
| Tokenize from the NLTK package is used in the CountVectoriser to generate word vectors 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 training dataset as both the validation and test sets are the unseen data for model validation.  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 vector for each tweet represents the frequency of a word from the vocabulary being in that tweet but not the order ( and therefore relationship) between the words in the tweet.  Normalisation: There are several techniques that can be applied to standardize the tokens in the vocabulary such as case folding all words to lower case or substituting emoji for text. Only folding to lower case was used here.  Lemmatizing the words to their root form (reading to read etc.) or steming words (use regex to drop prefixes & suffixes such as assuming an ‘s’ indicates a plural) can be useful, particularly in very large datasets, but they also remove some information. I have decided to exclude this step as the data set is small so does not require the simpler processing this gives.  The pre-processing of the 754-document training set reduced the vocabulary from 3349 to 3277 tokens. The compressed sparse row format of storing the word vectors reduced from 10510 to 8351 stored elements. | | | | | | | | | |
| **1.2 Model Evaluation** | | | | | | | | | |
| **Performance Metrics** | | | | | | | | | |
| The sckikit-learn metrics accuracy score, precision score, recall score and f1 score were used to evaluate the models. The classification report was used as this gives all the metrics above and the macro values ( value for each class). The weighted average is also given which is useful as it considers the support ( number of true instances for each label)  Precision is a useful metric for individual classes as it shows bias in the model. The macro average precision, weighted average precision and accuracy are poor metrics initially as the initial threshold for transforming the continuous sentiment scores into discrete values results in the labels being heavily biased towards positive. This imbalance in the size of classes means that if all classes were labelled as the dominant class these metrics would give an optimistic evaluation of the model.  Recall is a useful metric as it incorporates false negatives and highlights where less common classes are not catered for by the model. The macro average of the recall accounts for the specificity of the model by evaluating the models ability to label true negatives. This is more useful than the weighted average as it is not affected by imbalanced classes.  Confusion matrices were also used as this aids the understandability of the model more than just metrics by illustrating where the model is performing poorly. | | | | | | | | | |
| **Testing Procedure** | | | | | | | | | |
| Naïve Bayes was evaluated in addition to the chosen Logistic Regression model in order to give a baseline.  The same data split was used for each model and consisted of a training set with labels which was used to fit each model. A validation set which was used to for the initial validation of the model and a test set which was used to generate the final metrics & confusion matrices. Is this instance a validation set is not strictly necessary as the tuneable hyper parameters of the models are not needed for this task so there is no risk of overfitting the model if the test data is used.  Naïve Bayes , 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.  As can be seen from the metrics below, both models have similar overall accuracy scores ( due to the dominance of the positive sentiment class) but, with the exception of precision, the metrics for the Logistic model are more balanced across the classes.  The confusion matrices demonstrate that both models classify the majority of the data as positive with very poor performance in identifying True Positives for classes 0 & 1. | | | | | | | | | |
|  | | | | | | | | | |
| **Results** | | | | | | | | | |
| As can be seen from the metrics below, both models have similar overall accuracy scores ( due to the dominance of the positive sentiment class) but, with the exception of precision, the metrics for the Logistic model are more balanced across the classes.  The confusion matrices demonstrate that both models classify the majority of the data as positive with very poor performance in identifying True Positives for classes 0 & 1. | | | | | | | | | |
| **Class** | **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 |
| **Naïve Bayes without Data Processing** | | | | | | | | | |
| **Class** | **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 |
| **Logistic Regression Classifier without Data Processing** | | | | | | | | | |
| Applying the data processing and retraining the model gives improved metrics in all cases with the exception of Recall & F1 for the neutral class. The model now correctly classifies the majority of positive and negative data but there is still a significant bias to positively sentiment classification. | | | | | | | | | |
|  | | | | | | | | | |
| **Class** | **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 |
| **Logistic Regression Classifier with Data Processing** | | | | | | | | | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
| By adjusting the threshold scores the split of the data set is adjusted. With neutral sentiment being assigned to the continuous range between - 0.25 & 0.32, the dataset is split as follows: | | | | | | | | | |
| Negative labels: | | | | | | | | 283 | |
| Neutral labels: | | | | | | | | 359 | |
| Positive labels: | | | | | | | | 469 | |
| The actual proportion of positive, neutral and negative tagged tweets is unknown but the confusion matrix below indicates a realistic distribution. 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. | | | | | | | | | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
| **Class** | **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 |
|  | | | | | | | |  | |
| **Limitations** | | | | | | | | | |
| The Logistic Regression Classifier above is optimised as far as possible, it performs consistently across the classes with reasonable performance considering the complexity and subjectiveness of language.  In order to improve the process it would be necessary to validate the output of the model- the continuous value in the input data is ambiguous so the data classified by the model should be check by a human and the threshold adjusted accordingly.  A major failing of the current method is that it doesn’t account for context/ word order ( with the exception of the implemented negation function), a method such as skipgrams would allow more subtle sentiments to be accounted for.  Also, the current method doesn’t account for synonyms or idioms. A dictionary or lexicon would allow less common terms to be replaced with their more common synonym and so standardise the data to some extent. | | | | | | | | | |
|  | | | | | | | |  | |
| **1.3 Common Themes & Topics** | | | | | | | | | |
| First I identified the 10 most influential words in each class by extracting the words with the highest weighting coefficients for each classifier: | | | | | | | | | |
| Negative Sentiment: | | | | | Neutral Sentiment: | | | | Positive Sentiment: |
| ..... : 0.59  trouble : 0.60  back : 0.61  get : 0.62  bearish : 0.65  weak : 0.66  lower : 0.72  falls : 0.79  fall : 0.96  short : 1.45 | | | | | resistance : 0.57  options : 0.57  bbry : 0.60  may : 0.62  sells : 0.65  yhoo : 0.71  uk : 0.73  ipo : 0.77  today : 0.86  ceo : 0.90 | | | | calls : 0.67  ftse : 0.70  bullish : 0.70  britain : 0.73  highs : 0.76  hod : 0.76  higher : 0.76  strong : 0.86  buy : 0.97  long : 1.50 |
| To identify common topics positive and negative labelled data from the full dataset were divided into separate numpy arrays. Each array was preprocessed using the WordNet Lemmatizer to remove inflectional endings from the words in each document. This step also transforms the document using semantic relations so that the all words with the same meaning are replaced with the same synonym. This processing standardises the documents and reduces the size of the vocabulary which facilitates the analysis of common themes/ topics within the corpus.  The Dictionary function from the genism package is used to create the word to id mappings where each word in the vocabulary of the two corpus (positive & vocabulary) is given an id.  Each document from the two corpus is then converted into bag of words with the count of occurrences of each word id added to the corresponding location in the vector.  The LdaModel from the genism library is then applied to each corpus using a nominal value of 10 for the number of topics. The number of passes was also given a nominal value of 10. | | | | | | | | | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |
|  | | | | | | | |  | |