PR1: Medical Text Classification

Sunny Udhani
Student ID: 012457289
Rank: 11
F1-score: 0.7754

Approach

1. Preprocessing

- Fetch the training and test data from train.dat and test.dat files respectively to pandas dataframes.
- Separate the classes and medical text columns to independent lists which become our initial X_train and y_train and X_test.
- Initially lower the case of the statement so it is easy to process duplicates, Remove stopwords like ‘a’, ‘for’, ‘were’, etc from X_train and X_test. As well as remove lemmatizers from both. I used external library NLTK for removing stopwords and fixing lemmatizers.
- Pass the processed text from both X_train and X_test to build sparse matrices using TF-IDF score on the bag of words. Used parameterized sequential k-mer frequency representation of text and L2-norm of the matrix. Applied these strategies using TFIDFVectorizer.fit_transform on training data to build the vocabulary and TFIDFVectorizer.transform on test data to utilize that vocabulary.
- Fix the issue of an imbalanced dataset using imbalanced librarys’ oversampling technique SMOTE.
- Split the train data randomly using a random state seed so that when predicting using the test data we don’t get influenced by the imbalanced classes.

2. Classification

- Supply X_train, y_train and one row from the X_test for predicting y_test using cosine similarity and k-nearest neighbors. Perform a dot product for the X_train and x_test will give us cosine similarity as matrices are already normalized.
- Arrange the similar docs in a sorted manner based on the value.
- Find the classes of the first k similarities and store them in the counter data structure.
- Return the class with the highest count for y_test.
- Make a list of all the y_test values and write to the output.dat file.

Methodology

1. Preprocessing

- The first step in processing is to lowercase the statement and remove ‘english’ stopwords and punctuations because stopwords don’t add any meaning to the tf-idf calculation and are less important for similarity calculation against other medical words in the bag of words.
- Also, as the English language has many different same stem words I applied lemmatization and used them instead of the actual word because different statements may use the same word but in a different context resulting in different derivations.
- Then I used TFIDFVectorizer to translate the word list into a sparse matrix of L2 norm using n-range phrases(k-mer). TF-IDF is a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents.
- I used the fit_transform and transform methods of the vectorizer so that we build up vocabulary using the train data and calculate the values for the test data from the same as new unique words in the test data would only result to 0 in the dot product.
- Then as we have imbalanced data I used a sampling technique SMOTE to oversample under-represented classes and then use train_test_split to randomize the train data for better classification accuracy.

2. Classification

- For classification, I used cosine similarity as the similarity measure to find the neighbors for each test document. The program is very similar to the professor’s classify function of activity classification 1.
- I take the dot product of one document of the test matrix and the training matrix, which on sorting based on values gives us the most similar documents. We then find out the class of the k most similar documents of the train data and store these in the Counter data structure.
- We take the majority vote and return the class for the test row. On solving this for the all the records in the data, we get a list of classes predicted which are then written to the output file.
- I have parameterized the methods to take two meta-parameters for the classification model, k for the number of nearest neighbors and c for the k-mer text frequency taking into account words of phrases of length c.
- Achieved F1-score of 0.7754 for k=72 and c=2.