CA Assignment 2

Group#2

Avinash Kumar Chaurasia

Ambuj Kumar Mondal

Shashi Prakash

Saurabh Raut

30th May 2021

CA: Assignment 2 – Argument Mining
Group: #2

Approach:

- In the presented solution, we used the training data to train our model and verified against the validation data. We received a f1 score of 48.6 % on validation data and 85.7% on training data.
- Steps-
- We first loaded the training data into a data frame using the panda library. Then convert the complete text into lower case. We did the same with the validation data and created a data frame with lower cases.
- We have defined a function clean_words() where we get rid of special characters.
- This cleaning process is applied to the training data set provided.
- Extract features using vectorizing with nGrams.
- Identify Linguistic properties- Word count and sentence count.
- Use these features to create a classification model.
- Implement SVM for prediction.

CountVectorizing testing data and getting the features:

- We have counterVectorized the training_data on our refined text with ngram_range as (1,5) and max_features as 220.
- We came to the conclusion of choosing max_features to be 220 with ngram range = 1,5 after trying many combinations of max_features and ngram and getting the most optimal predictions on test data with these parameters.
- After getting vectors or word embeddings from the cleaned text, we created a data frame called training_features where we mapped the vectors with the cleaned text of training_data and give the defined the columns as vectors and hence, we got 2619 rows and 220 columns for training_features dataset. Training_labels were also created to assign labels from training data.
- Similarly, we have counterVectorized the validation data and created a data frame for validation_features.

Identifying certain linguistic properties for deciding the features:

- To identify certain linguistic properties, we have categorized the tokens on the basis of parts of speech i.e., verbs, adjectives and nouns.

- We have categorized the tokens on the basis of parts of speech as these verbs, adjectives and nouns have the strongest relationship while deciding the phrase as claim and non-claim and used these parts of speech as the features in our training_features and validation_features.

For ex.:


```
 {
        "id": "5f51e7274f0c92a9d3ed610cac6bb240",
        "text": "The freedom fighter/terrorist comparison is also an interesting
perspective.",
        "label": 1
    }


{
        "id": "47fa387ca94c32ab9489e65b000213c7",
        "text": "I still believe it is a good idea",
        "label": 1
    },
```


Most of the phrases having adjectives in them are more likely to be claims.

- Following the categorization, we have calculated the count for tokens on the basis of above linguistic properties.
- We have also added Number_of_sentences in a text, "word_count" in the refined text and Number_of_characters as new features for training as well as validation features Dataframe.

Putting these features as an input to the classification model:

- We have provided these features as an input to the support vector classifier and perform classification on the data set.
- We used a Support vector machine with kernel = poly and random_state = 0.
- We standardize the input data using StandardScaler() passed in the function make_pipeline. We used make_pipeline to enable the sequence of data to be transferred and correlated together in a model. It also avoids leaking of data.
- We have used GridSearch for finding the most optimal parameters to be passed for creating the model and found that C as 15 and kernel as 'Polynomial' provides the most optimal results.
- We fit the model with the training_features and labels and predict on the validation features.

Calculating F1 Score, precision and recall

- We have used the sklearn library on the generated metrices to calculate the precision, recall and F1 score.
- We got the F1 score as 0. 0.48666666666666664 for the validation data which surpasses the baseline value of 0.42.

Why have we used a Support vector machine for modelling and predicting the test data?

- We have tried to train our features and labels with other algorithms like Logistic Regression also, but the results were not very encouraging.
- One of the reasons which makes SVM better than other classifiers is the fact that it works on the principle of finding the best or maximum margin that separates the classes and hence, classifies the data in the most optimal way. On the other hand, Logistic regression tries to find the decision boundaries with different weights that are near to the optimal point.
- Furthermore, SVM also provides the feature of kernels which can convert the data in 2-dimensional space to higher dimensional space. Therefore, we can easily put the hyperplane between the classes and segregate the data. Logistic regression does not work well if the data is interleaved and it is difficult to draw hyperplanes in low dimension space, which was the case with us as we found a lot of our data points in training data were interleaved.
- We also had to avoid overfitting our model and as we know, the risk of overfitting is less in SVM while Logistic regression is vulnerable to overfitting.

Steps to run the code and generate predictions:

- We have produced a JSON file called output_file.JSON where we have mapped the Id's from validation_data to our predicted labels for validation data in the format of "<sentence_id_1>": <predicted label>.
- Then, we evaluate the f1-score of our model by running the eval.py file with command eval.py –true < location of validation_data.json > --predictions < location of output_file.json >. Please find below the screenshot, we got output values after running the eval.py file.