Classification of claims according to their veracity values

1 Text classification process

Text classification is a machine learning technique that assigns a set of predefined categories to open-ended text. Text classifiers can be used to organize, structure, and categorize pretty much any text from documents or emerging social media.

In order to complete this classification task, I basically divide the data processing process into the following six:

- Data Acquisition
- Text Cleaning
- Pro-Processing
- Feature Engineering
- Modeling
- Evaluation

My classification process has been shown in Figure 1.

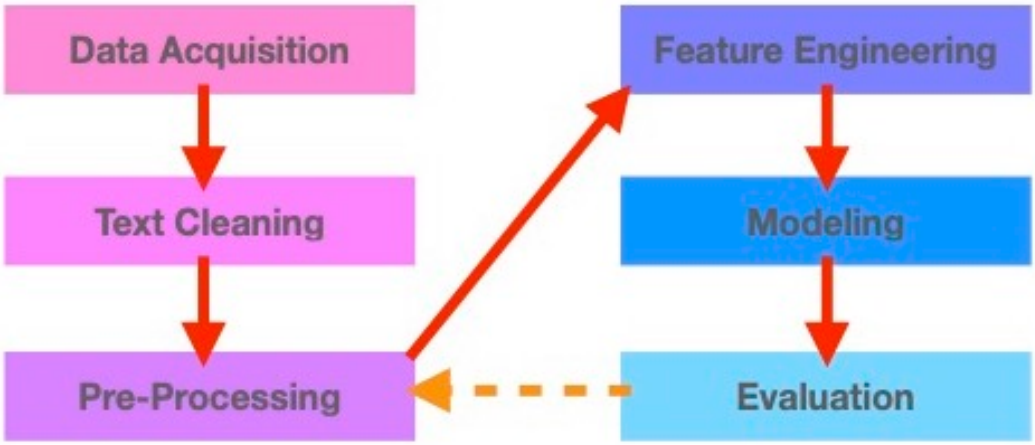

Figure 1: Text classification process

2 Data extraction

By observing the extracted data, I choose column `claimReview_source` as the target column for natural language processing and column `rating_alternateName` as the label for this classification learning.

However, because the values of column RATING__ALTERNATENAME are not uniform, I use the python code provided by webpage GitHub to re-categorize them to generate column NORMALIZEDRATINGS. And through the value of column NORMALIZEDRATINGS, generate column LABEL__2C and column LABEL__3C (corresponding to two different classification methods, respectively).

3 Tokenization

This time, I chose spaCy as a tool for text tokenization. The text is now pre-processed by reducing the lemmatized words (except for pronouns ('I')) and removing stop words. This process takes a long time. I only took the first data for an effect test and did not perform the operation on the entire database in the first. But this step will be completed in the process of model training.

The process of tokenization by using spaCy is as follows. (Figure 2)

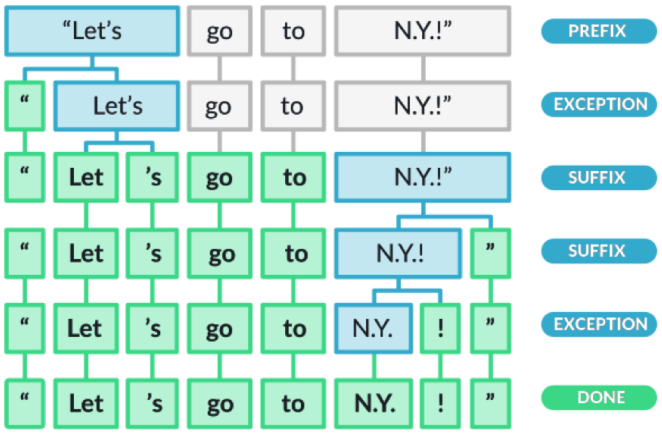

Figure 2: Tokenization by spaCy

4 Vectorization

In the part of vectorized features, I found that some documents have the same source and a large proportion and have great potential to have the same content theme (for example, politics), so I consider adopting the TF-IDF algorithm to vectorize. In order to be able to compare, I also added the bag-of-words algorithm. After all, not all documents have the same source and have the same theme. Perhaps the bag-of-words method is more appropriate for this example.

5 Test model

Originally, I planned to train the model through Bayesian linear regression and support vector machine methods, and compare the exact values and confusion matrix for further analysis. It is a pity that the actual operation time is too long, and the results have not been obtained. At present, my work can only stop here. I need to recheck my code in the future to determine whether I should make improvements. Unfortunately, I couldn't get the desired result and further analysis on this in the end.