CS4248 Assignment 2:

Solving fact checking

By A0155664X (Ong Jian Ying Gary)

*Note: Run assignment2.ipynb by putting the notebook in the location with train.csv and test.csv and run the entire notebook. The A0155664X.csv file will be generated in the same location as the final output predictions from the best model. It uses imblearn package so need to run !pip3 install imblearn*

**Exploratory Data Analysis (EDA)**

For the first part, we take in the raw data file and look at the distribution of labels in the training set. Illustration:

Chart, bar chart, treemap chart

Description automatically generated

We observe class imbalance whereby 65.26% belongs to ‘-1’, 24.06% belonging to ‘1’ and minority ‘0’ has only 10.68%. An attempt to improve class imbalance will be shown in preprocessing & model selection.

**Preprocessing**

Baseline model of LogisticRegession/MultinomialNB were used together with TfidfVectorizer/CountVectorizer with ngram\_range = (1,1), to assess whether we are met with improvement/deprovement of test f1\_score from prediction results on kaggle.

|  |  |
| --- | --- |
| **Preprocessing Technique** | **Worked/Did not work** |
| Casefolding & Data Normalization (MinMaxScaler) | Worked, helped with improving LogisticRegression (tfidf) f1\_score from 0.79682 to 0.8153 |
| Removal of duplicates and clipping of data with contradicting labels to minority class of 0 | Worked, helped model generalization of LogisticRegression (tfidf) from 0.8153 to 0.82531. |
| Lemmatization (considering POS tags) | Did not work |
| Removal of stopwords & punctuations | Did not work |

Case folding & data normalization helps the model probably because we are standardizing the letters to lowercase to reflect the same words, aside from named entities. MinMaxScaler() was especially helpful when dealing with counts of words using CountVectorizer, as some words frequency are higher than others. By data normalization, we are ensuring there are no scale-dominant feature that could lead to bias in the model. In addition, when new features are subsequently engineered, it is good practice to normalize the combined vectors from tfidf/counvec with new features so that scale is a non-factor.

Data normalization helped especially when using TfidfVectorizer, as it prevents importance of frequently occurring terms from impacting the model’s classification decision. We observed:

* LogisticRegression improve from 0.79682 🡪 0.8153
* NaiveBayes improve from 0.52517 🡪 0.76522

However, data normalization on the output of CountVectorizer was not as helpful because it considers only the frequency of each term, and it lacks term importance weighting and document length sensitivity.

* LogisticRegression deprove from 0.7922 🡪 0.7498
* NaiveBayes deprove from 0.81576 🡪 0.6874.

After looking at the data, it was observed that there are a few rows of data that are exact duplicates (same sentence, same label) and training data that are contradictory (same sentence, different label).

Graphical user interface, text, application

Description automatically generated

The contradictory data was first handled by clipping the values to the minority class label, so as to ensure better class balance during the training for generalization. In addition, we removed all exact duplicates of data in the training set as a preprocessing method to ensure that we do not overfit, induce bias into model, and avoid unnecessary computations. This proved to be true, as the test f1\_score of LogisticRegression (tfidf) improved from 0.8153 to 0.82531 and improved NaiveBayes model scores.

Lemmatization, removal of stopwords & punctuations did not help. One reason could be training data are short. Removal of stop words might lead to removal of important data. Hence, they were not used.

In summary, the **best baseline model score resulting from preprocessing ended up at from 0.82531.** We will improve on the score by performing manual feature engineering.

**Feature Engineering**

|  |  |
| --- | --- |
| **Type of Feature** | **Worked/Did not work** |
| Text Subjectivity / Text Polarity | Did not work, |
| Counts of different types of Part-of-Speech (POS) tags within a sentence | Worked, improved test f1 score from 0.82531 to 0.84159 for LogisticRegression (tfidf) |
| Dialog & Narrative parser | Worked |
| Unique words in sentence ratio | Worked, improved test f1 score from 0.84159 to 0.85446 |
| Stopwords in sentence ratio | Worked, improved test f1 score from 0.84159 to 0.85446 |
| Count of punctuations within the sentence | Did not work |

Text subjectivity refers to the degree of which a text expresses an opinion/personal view rather than information and text polarity refers to the sentiment orientation of a text. Inclusion of these features did not work for simple baseline models as MultinomialNB and LogisticRegression are linear models that learn to separate classes based on weighted sum of input features and might not be able to capture the full interplay between subjectivity and polarity. Hence, these were excluded from features to use.

Inclusion of counts of part-of-speech tags within a sentence helped with model prediction across the board for LogisticRegression & NaiveBayes because it provides grammatical structure and meaning of the sentence. By including this feature, the model can better understand the syntactic and semantic relationships between words in a sentence and the linguistic structures, improving the f1 score. Results from including POS tag counts:

* LogisticRegression (tfidf) improve from 0.82531 to 0.84159.

Inclusion of number of dialogs and narratives within a sentence can aid as they provide information about content of the text. Dialog refers to direct speech/conversation while narrative refers to a sequence of event in text. With these features, both LogisticRegression & MultinomialNB can learn to recognize patterns and type of statement.

Inclusion of unique words and stopwords in sentence ratio aided with model performance of test f1\_score. Unique words in sentence ratio reflects the diversity of vocabulary used in text, a text with high ratio may indicate more varied and sophisticated vocabular leading to a possible more factual statement, as opposed to a non-factual/unimportant statement. Similarly, the ratio of stopwords in sentence reflects the proportion of common function words in text. A text with high ratio may indicate more generic and impersonal language, while a text with higher ratio might lead to personal language. This is helpful for model generalization as both LogisticRegression and MultinomialNB improved f1\_scores, with the best scoring model being LogisticRegression with 0.85446 as the final score from 0.84159.

Inclusion of punctuations counts led to worse off as such feature may not be very informative. Punctuation marks are mainly used to separate words, indicate pauses or breaks in text rather than convey any special meaning. Hence, it led to poorer performance for both baseline models and hence, not used for final model.

In summary for feature engineering, only POS tagging counts, dialog & narrative parsing, and unique/stopwords ratio for each sentence had helped to boost the overall performance, **especially for LogisticRegrssion from a score of 0.82531 to a score of 0.85446, representing a ~2.9% increase in test f1\_score**.

**Model Selection & Hyper-parameter tuning**

In model selection, only the training data set was used to performed hyper-parameter tuning for different models and hyperparameters through 5-fold cross-validation, using GridSearchCV, for baseline models like MultinomialNB and LogisticRegression. Multilayer Perceptron (MLP) was also used but only attempted with a baseline setting. The training set is first split into 85% train and 15% validation to check validation f1 score. Both train & test data are preprocessed the same way according to the best results. Sampling methods such as oversampling (RandomOverSampler) was also trialed, to see if making up the minority class to have the same counts as the majority class would aid in model generalization. A ColumnTransformer object was utilized to perform vectorizing on ‘Text’ column only and MinMaxScaler() to scale all features between 0 to 1. For LogisticRegression, parameters to tune are ‘penalty’, ‘C’ which represents regularization extent. For MultinomialNB, parameter to tune is ‘alpha’ which represents extent of smoothing. For all models, TfidfVectorizer and CountVectorizer ‘ngram\_range’ are adjused to be (1,1), (1,2), (2,2), as it affects the range of ngrams to extract from data. Higher order n-grams might capture more context. CountVectorizers vect\_analyzer is varied between ‘word’ & ‘char’.

Table

Description automatically generated

Briefly, we can see that the top 3 models all belong to LogisticRegression, dominates the scores for validation f1\_score. LogisticRegression might be preferred over MultinomialNB because there is imbalanced dataset, and it might be able to adjust the decision threshold to better balance the trade-off between precision and recall.

For LogisticRegression models, we can see that l2-regularization penalty is preferred over l1-regularization. Ths could be due to the fact that l2-regularization might be able to handle correlated features better, by assigning small but non-zero weights to all features, rather than forcing some features to zero as in l1-regularization. L1-regularization might have been more appropriate if the dataset contains many unimportant and irrelevant features, which is not the case here.

The top 2 models having no oversampling involved. A simple duplication of minority data could have led to negative impacts, as there could be overfitting and model memorizes data instead of generalizing. A better way would be to augment new sentences using libraries like ‘nlpaug’ to aid with better f1\_scores, but they were not explored.

We can also see that across all results for CountVectorizer, word analyzer is preferred over characters as words capture higher level semantic meaning and are more interpretable.

Using the top 4 models of with highest validation f1\_score, the parameters were used to train on the full training dataset and used to predict test data labels and uploaded to kaggle. **Results show that the LogisticRegression (tfidf) with no oversampling performed the best out of all the models, with a final f1\_score of 0.85446, which can be replicated in the Jupyter notebook submitted.**

A simple multilayer perceptron (fully connected neural network) was also trialed. A total of 3 hidden layers (256,128,64 units), 1 input and output (3 units) layer with a softmax was used. L2-regularization was specified for each hidden layer to prevent overfitting. BatchNormalization layers were in between hidden layers to prevent problem of vanishing/exploding gradients during backpropagation & improves generalization due to similar data scales. EarlyStopping with patience of 3 was used to model validation loss to ensure no overfitting in training occurs.

A picture containing shape

Description automatically generatedChart, line chart

Description automatically generated

Training process showed no overfitting as the val\_loss & loss line are close to one another without great deviations. However, after running the test data sentences, the f1\_score was lacking at 0.67184 only, proving that a simple MLP cannot capture complex relationships due to limited capacity and has lack of consideration for word order as compared to other models like Recurrent Neural Networks. To improve scores, run hyper-parameter tuning for MLPs with more layers and varying units or use better models like RNN.

**Appendix**

1. **Declaration of Original Work**.

By entering my Student ID below, I certify that I completed my assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, I am allowed to discuss the problems and solutions in this assignment but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify my answers as per the Pokémon Go rule.

Signed, A0155664X

2. **References**.

I give credit where credit is due. I acknowledge that I used the following websites or contacts to complete this assignment

* <https://betterprogramming.pub/beginners-to-advanced-feature-engineering-from-text-data-c228047a4813>
* <https://towardsdatascience.com/nlp-with-pipeline-gridsearch-5922266e82f4>