Queen Mary, University of London
School of Electronic Engineering and Computer Science

ECS763P - Natural Processing Language

Sequence Classification

Mughees Asif | 180288337

Due: 22 November 2021

Question 1

The data was loaded and trained using no preprocessing for this section. The data was then split into training and testing sets by using array slicing. The dataset contained a total of 7816 samples, where after the split, the `train_data` contained the first 80% of the total samples (6252 samples), while the `test_data` was 20% (1564 samples). This method also preserved the state of the input data without any randomness throughout the course of the project.

Question 2

Table 1: Top 5 classes with the lowest precision

Class	Score
I-Opinion	0.071111
B-Plot	0.215287
B-Origin	0.365239
B-Opinion	0.432584
B-Soundtrack	0.437500

The sentences associated with `false positives` were printed by looping over the `predictions` array which was filled with the original dataset sentences broken into words and tagged by the `CRFTagger` from the `NLTK` package. To print the five lowest false positive classes, the classification report was converted into a `pandas.DataFrame` which was then sorted in an ascending hierarchy for precision, as displayed in Table 1.

Question 3

Table 2: Top 5 classes with the lowest recall

Class	Score
I-Soundtrack	0.069705
B-Soundtrack	0.070707
I-Character_Name	0.090909
B-Quote	0.092199
I-Opinion	0.092843

The sentences associated with `false negatives` were sourced in a similar way as in Q2, above. To print the five lowest false negative classes, the classification report was sorted in an ascending hierarchy for recall, as displayed in Table 2.

Question 4

The `CRFTagger` was used to add part-of-speech `POS` tags to the dataset. The dataset was first split using a different variation of preprocessing, where the `POS` and the extracted word were concatenated with the operator to differentiate between the two. Thereon, the `get_features` function was changed to split the incoming tagged word at the operator to append the appropriate feature to the correlating word. The features were then used to train the tagger to recognise

the POS tagged words that were then analysed using the classification report. The comparison highlighted a marked improvement in all macro average metrics with the net increases displayed in Table 3.

Table 3: Comparison of macro average scores with and without POS tags

Metric	Without POS tags	With POS tags	% increase
Precision	0.642763	0.777209	21
Recall	0.457124	0.608033	33
F1-score	0.493850	0.653593	32

Question 5

The selection of relevant features was a laborious process as a lot of different variations were tried to optimise the macro weighted averages. The same process from Q1–Q4 was repeated to prepare the dataset for training. The `get_features` function was changed to incorporate 7 more POS tags, as highlighted in the notebook. Furthermore, during the tagging, two additional hyperparameters were also added: the minimum frequency of features was changed from 1 to 2, and $L2$ regularisation with a value of 0.1 (chosen after referring to online literature). The minimum frequency was changed to reflect the increase in the POS tags, and the $L2$ regularisation was introduced to reduce overfitting. As the POS tags and features were increased incrementally, the metrics also reflected a percentage increase highlighted in Table 4.

Table 4: Effect of adding features

Metric	Normal features	Extended features	% increase
Precision	0.777209	0.868984	12
Recall	0.608033	0.791476	30
F1-score	0.653593	0.824744	26

Introducing and implementing relevant features increased the tagger's capacity to learn and produce more accurate predictions. From the start to the end of the project, the macro weighted metrics slowly increased with the addition of new features. Precision, recall and the F1-score increased by 35%, 79%, and 67% as compared to the vanilla preprocessing and feature extraction from Q1.