NLP – Assignment 1

Mitchell Caridad Soluren Jr
2006-25306

Question 1

The first step to this question was to do a deterministic split where the first 80% of the original training set was retained for model training while the rest became the validation set.

A conditional random field (CRF) was then trained on the new training set, and its performance was determined on the validation set. Based on the classification report of the latter, it was shown that the macro f1 score of the initial model was 55%.

Through the process above, it was noted that the data contained no capitalizations and punctuations, indicating that it was probably pre-processed beforehand. Moreover, the data was also unbalanced, as some classes like "B-Soundtrack" had less than 50 tokens in the validation set while others had upwards of a thousand. Moreover, most of these "minority" classes had the worst f1 scores, the only exceptions being "I-Year" and "B-Award".

Questions 2 and 3

These are error analysis questions, where the challenge is that tagging is done on a token-level but mispredictions need to be printed at the sentence-level. The first step to answering them was to generate the classes with the worst precision and recall scores on the validation set. These were:

Classes with the worst precision scores

I - Soundtrack	B - Soundtrack	I - Opinion	B - Opinion	B - Plot

Classes with the worst recall scores

B - Soundtrack	I - Soundtrack	I- Opinion	I- Character_ Name	B- Character_ Name

Some functions were then written to generate token and sentence-level data frames for a specified array-form dataset and crf model. These were then used to print out sentences with false positives, false negatives, and correct predictions based on a specified list of classes. The print style of these functions was formatted so that the following elements were immediately visible: the sentences, the true and predicted tags per token, and the target FP or FN tokens.

Through these functions, it was observed that model made no predictions for the "Soundtrack" classes, but one possible way to remedy this is to add backward-window word features as the word "song" usually occurs before a song name. Moreover, it was noted that some of the misclassified tokens in the "Opinion" classes were modifiers like adjectives and adverbs, hence the addition of POS tag features may help improve the model's performance on these classes. Similarly, the model misclassified determiners like "a" as "B-Plot" tokens, so the addition of POS tags and backward-window features could help mitigate these errors. Finally, "Soundtrack" classes, "Opinion" classes, and "B-Plot" were found to possibly benefit from the use of gazetteers, but the implementation may be difficult. Moreover, the gazetteers could adversely affect the other classes.

Question 4:

This question required the addition of POS tags as features of the CRF. To do this, the pre-process function was modified so that a pre-built POS-tagger CRF model is run on the sentence to be pre-processed. From here, each token in sentence is then concatenated to its corresponding POS tag, with the output being of the form *[(word_1@pos_1,ner_1), ...(word_n@pos_n,ner_n)]*. From here, the *get_features* function was then modified so that it extracts POS tags using the *split()* function. These are then added to the feature vector of a particular word in a sentence.

After doing these modifications, the crf model was retrained and it was found that its f1 macro-score on the validation set increased from 55% to 56%. It was also shown that "B-Opinion", "I-Opinion", and "B-Plot" got slight increases in their precision scores. However, the model still made no "Soundtrack" predictions. Moreover, of the five worst recall classes, only "I-Character_Name" got an increase in recall score.

Question 5

This question involved using feature engineering, feature selection and hyperparameter tuning to optimize the CRF's macro-f1 score. To do so, a modelling methodology was formulated, as in section 5.1 of the notebook. This ensured that modelling decisions were anchored to an analytical workflow.

It was also decided that capitalization and punctuation would be removed as features because the data does not contain them. Instead, backward and forward-window features were introduced, along with variable-length suffixes and prefixes. Window features are the features of the words beside the current word. Moreover, it was decided that the minimum feature frequency would be set to 5 for the feature selection phase, as this hyperparameter helps remove possibly unnecessary features from the onset. Moreover, this would still be tuned during the hyperparameter tuning phase.

For model iteration 1, the chosen features were the window features of length 3 (i.e. features of the 3 words before and 3 words after the current word), POS tags, and suffixes and prefixes up to length 3. The results showed an immediate improvement versus the POS-only model, with the macro f1 score increasing to 65%. However, the training set performance showed that the model was overfitting, as the macro-f1 score on the training set was 88%. This meant that some features needed to be dropped.

Aside from performance, this model iteration also helped confirm some the observations made in questions 2 and 3, as it was noted that the back-window features for the word "song" helped predict the "Soundtrack" classes, while the adjective and adverb features helped predict the "Opinion" classes. However, it was also noted that the determiner feature helped predict the "B-plot" class, which was the opposite of what was expected. These findings translated to performance gains in the previous 2 class families and a performance drop in "B-plot", as seen in section A.7 of the notebook.

For model iteration 2, it was decided that the window lengths would be shortened to 2. Moreover, some iterative feature selection showed that setting suffix length to 1 increased performance. Hence the feature set for iteration 2 was suffixes up to length 1, prefixes up to length 3, and window features of length 2. This change resulted in a macro f1 score of 67% in the validation set, and performance gains in all the worst-predicted classes. Moreover, difference between training and test set performance was lower, indicating that the overfit had decreased.

Model iteration 3 involved the use of *name.basics.tsv.gz* dataset from IMDB, which is a public database that contains data on movie crew members from different films in the IMDB data repository (IMDb, 2021), to create gazetteer features for actors, directors, and composers. This involved a 2-token window search function to check if a pair of sequential tokens are in a list of actor, director, and composer names generated from the IMDB dataset. However, doing this resulted in a very long run time and hence the actor list was dropped. Moreover, the resulting model did worse than model iteration 2, hence the use of gazetteers was dropped.

Afterwards feature selection, the next step was to use Bayesian optimization through the *GP_minimize* function of scikit optimize to find the optimal hyperparameters. This function was chosen because it is ideal for models with long training times (Head T., 2018), and the process involved defining a feature space and an objective function that computed negative macro-f1 score on the validation set. The resulting performance of the tuned model on the validation set was 68%.

From here, the model was retrained on the full training set and the final summary is as follows:

Final Features	Hyperparameters	Macro F1 score (val. Set)	Macro F1 score (test set)
- Words - POS tags - all suffixes up to length 1 - all prefixes up to length 3 - number indicator All with window [-2.2]	c1=0.471216 c2=0.473724	68%	65.9%~ 66%

Sources:

Head T., M. G. e. a., 2018. gp_minimize/scikit-optimize: v0. 8.1. [Online]
Available at: https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html
[Accessed November 2021].

IMDb, 2021. name.basics.tsv.gz/IMDb Datasets. [Online]
Available at: https://www.imdb.com/interfaces/
[Accessed November 2021].