**TASK 1**

In this individual project, I adopted a multi-dimensional experimental strategy to optimize the test accuracy of the model. Before starting, I made some changes to the code, here are the declarations:

1. **Punctuation Handling**: Initially, I disabled the 'remove punctuation' feature to assess its impact, treating it as a tuneable parameter.
2. **Early Stopping Adjustment**: To expedite multiple runs, I reduced the early stopping criterion from 5 to **3** epochs and added a condition to halt training if training accuracy reached 99.9% to prevent memorize.
3. **Word Embedding**: use glove-wiki-gigaword-300 as it captures rich semantic relationships, making it more effective for sentiment classification.
4. **Original test result**: **test accuracy = 86.52%, loss = 0.352** with pre-adjust code, and other parameters are 1) Frequency cutoff = 25, 2) Hidden dim = 20, 3)Batch size=64, 4)Learning rate = 0.001, 5)One-hot vectorize.

STEP 1: Given the numerous adjustable parameters, testing all possible combinations is impractical. Therefore, I adopted a greedy algorithm approach, first analysing the impact of individual parameter changes to identify potential trends and then conducting deeper tests on selected parameters to achieve a local optimal solution. The table below presents the test accuracy and loss for each parameter modification:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Action | Acc | Loss |  | No. | Action | Acc | Loss |
| 1 | Remove punctuation | 84.38 | 0.361 |  | 2 | Remove special characters | 84.38 | 0.372 |
| 3 | Case folding | 86.13 | 0.354 |  | 4 | Expand contractions | 85.64 | 0.351 |
| 5 | Remove stop words | 82.51 | 0.402 |  | 6 | Wordnet Lemmatize | 85.64 | 0.349 |
| 7 | Frequency cutoff (0) | 87.04 | 0.303 |  | 8 | Frequency cutoff (50) | 84.08 | 0.367 |
| 9 | Hidden dim (10) | 86.33 | 0.337 |  | 10 | Hidden dim (30) | 85.54 | 0.349 |
| 11 | Batch size (64) | 86.31 | 0.344 |  | 12 | Batch size (128) | 86.52 | 0.347 |
| 13 | Learning rate(0.0001) | 86.71 | 0.355 |  | 14 | Learning rate(0.01) | 83.40 | 0.407 |
| 15 | Term frequency | 86.32 | 0.331 |  | 16 | TF-IDF (ngram:1-1) | 88.76 | 0.262 |
| 17 | TF-IDF (ngram:1-2) | 90.14 | 0.284 |  | 18 | Opinion lexicon | 86.91 | 0.327 |
| 19 | Word embedding | 85.45 | 0.350 |  | 20 | Additional hidden layer | 86.52 | 0.354 |
| 21 | Dropout rate(0.25) | 86.62 | 0.344 |  | 22 | Dropout rate(0.5) | 86.13 | 0.346 |
| 23 | Batch normalization | 83.88 | 0.368 |  | 24 | Weight decay | 86.72 | 0.351 |

Based on the table above, I categorized the parameter modifications into three groups:

1. Retention (Green-marked): the parameters highlighted in green improved test accuracy compared to the base line.
2. Retention (Yellow-marked, considered as acceptable variation) : the parameters in yellow showed test accuracy close to the baseline, suggesting that the observed differences might be within an acceptable error range.
3. Rejection (Unmarked, no improvement): 1) parameters that did not improve or even reduced test accuracy will not be retained. 2) additionally, modifying hidden dim and batch size (whether increase or decrease) had negligible impact on test accuracy, so these will also be excluded from further tuning.

STEP 2: Since the number of parameters still large, I used **Optuna** (n=50) to automate hyperparameter tuning. This approach efficiently explores different configurations, helping to identify the best-performing model while balancing generalization and overfitting. The top 10 results were extracted for further analysis.