**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 train accuracy reached 99.9% to prevent the model from memorizing the entire dataset.
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 chars | 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. In addition, to *avoid getting stuck* in a local optimum, I chose *optuna.samplers.RandomSampler(),* which selects hyperparameters randomly instead of relying on past trails.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Top | Vect. type | Freq cut. | LR | Case fld. | N-gram | Op. lex. | Add. layer | Drop-out | W. decay | Acc (%) |
| 1 | TF-IDF | 23 | 0.00021 | True | (1, 2) | False | True | 0.13 | 0.00001 | 91.50 |
| 2 | TF-IDF | 21 | 0.00056 | False | (1, 2) | False | True | 0.12 | 0.0001 | 91.50 |
| 3 | TF-IDF | 22 | 0.00018 | False | (1, 2) | False | True | 0.18 | 0 | 91.40 |
| 4 | TF-IDF | 23 | 0.00012 | True | (1, 2) | False | True | 0.21 | 0 | 91.30 |
| 5 | TF-IDF | 5 | 0.00037 | False | (1, 2) | False | True | 0.2 | 0.0001 | 91.21 |
| ……like above…… | | | | | | | | | | |
| 19 | TF-IDF | 2 | 0.0001 | False | (1, 1) | True | True | 0.22 | 0 | 89.65 |
| 20 | TF-IDF | 20 | 0.0001 | False | (1, 1) | True | True | 0.18 | 0 | 89.64 |
| ……. | | | | | | | | | | |
| 27 | Term Freq | 3 | 0.00026 | False | Nan | True | False | 0.19 | 0 | 88.77 |
| 28 | Term Freq | 0 | 0.00025 | False | Nan | True | True | 0.22 | 0 | 88.67 |
| ……. | | | | | | | | | | |

The results clearly show that TF-IDF with Ngram (1, 2), opinion lexicon disabled, and an additional layer consistently outperforms all other combinations, achieving the highest accuracy. Trying IF-IDF with higher N-gram settings, such as (1, 3), may yield even better results.

**STEP 3**

Manually adjust the hyperparameters.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trail | | Ngram | Freq cut. | LR | | | | Case fld. | | Drop-out | W. decay | | Acc (%) |
| 1 | | (1, 3) | 23 | 0.00021 | | | | True | | 0.13 | 0.00001 | | 86.8 |
| 2 | | (2, 3) | 23 | 0.00021 | | | | True | | 0.13 | 0.00001 | | 89.54 |
| Increasing the N-gram value did not improve accuracy. Stop the current trial and focus on finding better accuracy using (1,2) | | | | | | | | | | | | | |
| 3 | | (1, 2) | 23 | 0.00021 | | | | False | | 0.13 | 0.00001 | | 91.30 |
| 4 | | (1, 2) | 23 | 0.00021 | | | | True | | 0.13 | 0 | | 91.11 |
| 5 | | (1, 2) | 22 | 0.00021 | | | | True | | 0.13 | 0.00001 | | 91.30 |
| 6 | | (1, 2) | 30 | 0.00021 | | | | True | | 0.13 | 0.00001 | | 91.31 |
| 7 | | (1, 2) | 50 | 0.00021 | | | | True | | 0.13 | 0.00001 | | 91.31 |
| 8 | | (1, 2) | 50 | 0.0001 | | | | True | | 0.13 | 0.00001 | | 91.31 |
| It appears that the current results have at least reached a local optimum. Therefore, I decided to reintroduce previously discarded variables to explore whether there is still an opportunity for further improvement. The following experiments are based on the Top 1 parameters from Step 2, testing the effects of Remove punctuation, Remove special characters, Expand contractions, Remove stop words, Wordnet Lemmatization, Word embedding, and Batch normalization. | | | | | | | | | | | | | |
| No. | Action | | | | Acc |  | No. | | Action | | | Acc | |
| 9 | Remove punctuation | | | | 90.33 | 10 | | Remove special chars | | | 90.52 | |
| 11 | Expand contractions | | | | 90.82 | 12 | | Remove stop words | | | 90.53 | |
| 13 | Wordnet Lemmatize | | | | 91.99 | 14 | | Word embedding | | | 90.04 | |
| 15 | Batch normalization | | | | 90.92 |  | |  | | |  | |
| Fortunately, enabling Wordnet Lemmatization further improved accuracy. Next, I will explore combining it with TF-IDF (1,1) and Term Frequency to see if it can yield even better results. | | | | | | | | | | | | | |
| 16 | Wordnet Lemmatize with Top 19 TF-IDF (ngram: 1,1) | | | | 88.18 |  | 17 | | Wordnet Lemmatize with Top 20 TF-IDF (ngram: 1,1) | | | 88.96 | |
| 18 | Wordnet Lemmatize with Term Freq (Top 27) | | | | 88.38 | 19 | | Wordnet Lemmatize with Term Freq (Top 28) | | | 89.26 | |
| Through multiple rounds of manual experiments, I unexpectedly discovered that applying Wordnet Lemmatization not only improves the performance of TF-IDF but also has a positive impact on Term Frequency. | | | | | | | | | | | | | |

**Final**

Based on multiple trails in step 3, I extracted the top 5 model parameters, as detailed below

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Top | Vect. type | Freq cut. | LR | Case fld. | N-gram | Op. lex. | Add. layer | Drop-out | W. decay | Acc (%) |
| 1 | TF-IDF (with Wordnet Lemmatize) | 23 | 0.00021 | True | (1, 2) | False | True | 0.13 | 0.00001 | 91.99 |
| 2 | TF-IDF | 23 | 0.00021 | True | (1, 2) | False | True | 0.13 | 0.00001 | 91.50 |
| 2 | TF-IDF | 21 | 0.00056 | False | (1, 2) | False | True | 0.12 | 0.0001 | 91.50 |
| 3 | TF-IDF | 22 | 0.00018 | False | (1, 2) | False | True | 0.18 | 0 | 91.40 |
| 4 | TF-IDF | 30 | 0.00021 | True | (1, 2) | False | True | 0.13 | 0.00001 | 91.31 |

**Conclusion**

In this sentiment classification task, most preprocessing techniques, except for Wordnet Lemmatization, showed little to no improvement, with some even having a negative impact. Among various vectorization methods, TF-IDF demonstrated a clear advantage, and a well-chosen unigram + bigram combination also contributed positively. As for other hyperparameters, the main challenge lies in striking the right balance to achieve optimal performance. In practice, hyperparameter tuning is not a linear process, and the interactions between multiple parameters are not entirely independent, making it even more challenging to identify the optimal combination.

**TASK 2**

Before starting, here are the **declarations**:

1. Similar to Task 1, I reduced the early stopping criterion from 5 to **3** epochs and added a condition to halt training if training accuracy reached 99.9%.
2. Original test result: test accuracy = 50.1%, loss = 1.70, and other parameters are 1)Hidden dim = 300, 2)Batch size=64, 3)Learning rate = 0.001, 4)One-hot vectorize, 5) dropout rate=0.2, 6) batch normalization.

**STEP 1**

Similar to Task 1, I adopted a greedy algorithm approach and tested individual parameters to identify that worth further optimization.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Action | Acc | Loss |  | No. | Action | Acc | Loss |
| 1 | Hidden dim (200) | 46.5 | 1.75 |  | 2 | Hidden dim (400) | 53.1 | 1.67 |
| 3 | Learning rate(0.0001) | 46.1 | 1.77 |  | 4 | Learning rate(0.01) | 48.1 | 1.82 |
| 5 | Batch size (32) | 50.4 | 1.69 |  | 6 | Batch size (128) | 49.5 | 1.69 |
| 7 | Dropout rate (0) | 50.5 | 1.70 |  | 8 | Dropout rate (0.3) | 51.6 | 1.72 |
| 9 | Case folding | 42.4 | 1.84 |  | 10 | Additional hidden layer | 48.6 | 1.72 |
| 11 | Term frequency | 52.6 | 1.65 |  | 12 | TF-IDF (ngram:1-2) | 65.4 | 1.35 |
| 13 | Batch norm (False) | 44.2 | 1.75 |  | 14 | Weight decay (0.0001) | 49.8 | 1.70 |

The results of this experiment differ significantly from those in Task 1, indicating that changes in *Hidden dim* and *Batch size* had an impact on performance. Additionally, increasing or decreasing the *Dropout rate* both led to improvements in test accuracy, suggesting that this parameter may introduce considerable variability in model performance. Notably, using *TF-IDF (ngram:1-2)* resulted in a substantial accuracy increase to 65.4%, highlighting the significant influence of n-gram selection on model performance. Therefore, further exploration and optimization in this area are necessary.

**STEP 2**

Similar to Task 1, I also used *Optuna* to conduct 50 random trails to explore the best hyperparameter configurations. This approach ensures a broader search space, increasing the chances of finding better-performing configurations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top. | Vectorizer | Ngram | Hidden dim | Batch size | Dropout rate | Weight decay | Test Acc |
| 1 | TF-IDF | (2, 4) | 320 | 8 | 0.41 | 0 | 71.32 |
| 2 | TF-IDF | (1, 3) | 300 | 32 | 0.24 | 0.0001 | 71.08 |
| 3 | TF-IDF | (2, 3) | 410 | 8 | 0.23 | 0 | 70.47 |
| 4 | TF-IDF | (2, 3) | 370 | 32 | 0.14 | 0.0001 | 70.28 |
| 5 | TF-IDF | (2, 4) | 380 | 8 | 0.38 | 0.0001 | 70.23 |
| 6 | TF-IDF | (1, 3) | 480 | 32 | 0.5 | 0.00001 | 69.67 |
| 7 | TF-IDF | (2, 4) | 350 | 16 | 0.29 | 0.00001 | 69.35 |
| 8 | TF-IDF | (3, 4) | 460 | 16 | 0.01 | 0.00001 | 68.14 |
| 9 | TF-IDF | (1, 2) | 300 | 64 | 0.11 | 0 | 68.06 |
| 10 | TF-IDF | (1, 2) | 390 | 64 | 0.22 | 0.00001 | 67.56 |
| …… | | | | | | | |
| 13 | TERM | Nan | 310 | 16 | 0.04 | 0.0001 | 55.16 |
| 14 | TERM | Nan | 410 | 16 | 0.31 | 0.0001 | 53.64 |
| 15 | TERM | Nan | 380 | 16 | 0.21 | 0 | 53.58 |
| 16 | ONE-HOT | Nan | 460 | 8 | 0.07 | 0.00001 | 53.56 |
| …… | | | | | | | |

Based on above results, TF-IDF consistently outperforms other vectorization methods, indicating that further exploration within TF-IDF could yield even better test accuracy. Additionally, the top performing sets tends to have lower hidden dimensions, suggesting that reducing the hidden dimension could be beneficial. Another trend is that smaller batch size generally lead to higher accuracy, implying that a lower batch size might be preferable, making it an area worth further fine-tuning.

Above all, I will focus on refining TF-IDF based configurations while experimenting with reduce hidden dims, smaller batch size, and minimal weight decay to optimize test accuracy.

**STEP 3**

Manually adjust the hyperparameters.

* 1. Trail based on Top 1 (Ngram=(2, 4)).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trail | Hidden dim | Batch size | Dropout | Weight decay | Acc | Notes |
| Baseline | 320 | 8 | 0.41 | 0 | 71.32 | Init best config |
| 1 | 300 | 8 | 0.41 | 0 | 70.04 | Lower dim reduced accuracy |
| 2 | 340 | 8 | 0.41 | 0 | 72.46 | Increasing dim improved accuracy |
| 3-4 | 360 → 330 | 8 | 0.41 | 0 | 70.65 → 71.44 | Dim=340 is best. Freeze dim. |
| 5 | 340 | 16 | 0.41 | 0 | 70.00 | No improvement. |
| 6-12 | 340 | 8 | 0.2→0.45 | 0 | 70.89 → 72.40 | Change dropout rate no improvement. |
| 13 | 340 | 8 | 0.41 | 0.0001 | 70.83 | No improvement. |

* 1. Trail based on top 2 (Ngram=(1, 3)).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trail | Hidden dim | Batch size | Dropout | Weight decay | Acc | Notes |
| Baseline | 300 | 32 | 0.24 | 0.0001 | 71.08 | Init best config |
| 1 | 320 | 32 | 0.24 | 0.0001 | 69.73 | Increasing dim reduced accuracy |
| 2 | 280 | 32 | 0.24 | 0.0001 | 70.4 | Lower dim reduced accuracy |
| 3 | 290 | 32 | 0.24 | 0.0001 | 69.11 | No improvement, Freeze dim. |
| 4 | 300 | 16 | 0.24 | 0.0001 | 72.08 | Improve accuracy. |
| 5 | 300 | 8 | 0.24 | 0.0001 | 70.47 | Freeze batch size to 16. |
| 6-15 | 300 | 16 | 0.2-0.3 | 0.0001 | 71.96→  72.27 | When dropout rate=0.28, get highest accuracy. |
| 16 | 300 | 16 | 0.28 | 0.00001 | 71.84 | No improvement. |
| 17 | 300 | 16 | 0.28 | 0 | 71.29 | No improvement. |

* 1. Trail based on Top 3 (Ngram=(2, 3))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trail | Hidden dim | Batch size | Dropout | Weight decay | Acc | Notes |
| Baseline | 410 | 8 | 0.23 | 0 | 70.47 | Init best config. |
| 1 | 350 | 8 | 0.23 | 0 | 69.6 | No improvement. |
| 2 | 400 | 8 | 0.23 | 0 | 70.35 | No improvement. |
| 3 | 420 | 8 | 0.23 | 0 | 68.78 | Freeze dim to 410. |
| 4 | 410 | 16 | 0.23 | 0 | 69.23 | No improvement, freeze batch size to 8. |
| 5-14 | 410 | 8 | 0.2-0.3 | 0 | 70.47→  70.89 | When dropout rate=0.24, get highest accuracy. |
| 15 | 410 | 8 | 0.24 | 0.00001 | 70.23 | No improvement. |

**Final**

Based on multiple trails in step 3, I extracted the top 5 model parameters, as detailed below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top. | Vectorizer | Ngram | Hidden dim | Batch size | Dropout rate | Weight decay | Test Acc |
| 1 | TF-IDF | (2, 4) | 340 | 8 | 0.41 | 0 | 72.46 |
| 2 | TF-IDF | (2, 4) | 340 | 8 | 0.42 | 0 | 72.40 |
| 3 | TF-IDF | (1, 3) | 300 | 16 | 0.28 | 0.0001 | 72.27 |
| 4 | TF-IDF | (1, 3) | 300 | 16 | 0.24 | 0.0001 | 72.08 |
| 5 | TF-IDF | (1, 3) | 300 | 16 | 0.28 | 0.00001 | 71.84 |

Note: the above results all based on same Learning rate (0.0001), no case folding, no additional hidden layer, no batch normalization.

**Conclusion**

In this surname classification task, TF-IDF once again proved to have a clear advantage as a vectorizer over other methods. However, just like in Task 1, finding an optimal combination of hyperparameters remains a significant challenge.