Bag of N-Gram Document Classification

Mingsi Long | ml5893 | https://github.com/ml5893/NLP_A1

1 Base Model

We adopt Bag of 2-Gram as the base model.

- Tokenization schemes of the dataset

  For a given review $r =$ "$a \quad b \quad c \quad d! \quad haha \quad :)$", first we split it by spaces and punctuations, and deleting punctuations, *r1 = ["a", "b", "c", "d", "haha"] . Then we use ngram from package nltk to tokenize it into 1-gram tokens and 2-gram tokens. After tokenization, the reviews is r2 = [("a"), ("b"), ("c"), ("d"), ("haha"), ("a", "b"), ("b", "c"), ("c", "d"), ("d", "haha")].*

  There are total 9481452 *tokens in the train data set.*

- *Model hyperparameters*

  The vocabulary size is 20000*. Each token in the vocabulary appears more than* 37 *times in the train data set. The embedding size is* 200*.*

- *Optimization hyperparameters*

  The optimization is Adam. The learning rate is 0.0005*. Here we don't use annealing learning rate.*

- *Other hyperparameters*

  The batch size is 36*. The maximum token number of each review is* 200*. The running epoch is* 10*.*

The accuracies on validation data is 87.04, the accuracy on test data is 86.168.

2 Experiments with different hyperparameters

2.1 Tokenization schemes

The all hyperparameters are the same except not deleting punctuations when tokenizing reviews. We compare it to the base model in the following table.

Table 1: Performance Comparison

	tokens in train	count barrier	validation accuracy	test accuracy
base model	9,481,452	37	87.04	86.168
model with punctuations	11,265,748	44	86.66	85.464

The training curves of these two models are plotted in Figure 1. We can see that tokenization without punctuations is better. Punctuations appears more times but have less sentiment information.

2.2 Model hyperparameters

We vary n for $n-gram$ (n=1,2,3,4) and the other hyperparameters are the same as base model. The results are listed in the following table.

Table 2: Performance Comparison

n	all	train tokens	count barrier	validation accuracy	test accuracy
1	true	$4,750,726$	8	87.88	85.812
2	true	$9,481,452$	37	87.04	86.168
3	true	$14,192,178$	48	86.88	85.972
4	true	$18,882,904$	50	86.26	86.116
2	false	$4,730,726$	25	85.22	84.844
3	false	$4,710,726$	14	80.15	80.028
4	false	$4,690,726$	6	72.92	73.368

The training curve is plotted in the Figure 2. We can see that when if n-gram doesn't contain the n'-gram such n'$<$ n, when n grows, the performance becomes worse. This comes from that when n grows, the tokens number decreases.

Varying the vocabulary size, the results are listed as following. The training curve is plotted in Figure 3.

Table 3: vocabulary size

vocabulary size	count barrier	validation accuracy	test accuracy
5000	164	85.38	85.508
10000	80	86.16	85.972
20000(base)	37	87.04	86.168
30000	25	87.40	85.732

We can see that the performance is better when vocabulary size grows but the training time grows.

Varying the embedding size, the results are listed as following. The training curve is plotted in Figure 4

Table 4: embedding size

embedding size	running time	validation accuracy	test accuracy
5	3min	73.58	71.912
50	4min	85.56	84.736
100	6min	86.62	85.996
200(base)	10min	87.04	86.168
400	12min	87.24	85.812

We can see that when embedding size grows, the performance on validation set grows but it's not necessary on test data set, since large embedding size to make model overfitted.

2.3 Optimization hyperparameters

Varying the optimizer from Adam to SGD, the results are as listed. The training curve is plotted in Figure 5.

Table 5: Performance Comparison

Optimizer	validation accuracy	test accuracy
Adam(base)	87.04	86.168
SGD	54.98	54.566

Varying the learning rate, the results are listed as following. The training curve is plotted in Figure 6.

Table 6: Performance Comparison

learning rate	decay function	validation accuracy	test accuracy
0.01	None	83.12	80.616
0.0005(base)	None	87.04	86.168
0.0001	None	78.28	78.392
0.001	$(0.8)^{epoch}$	87.18	86.012
0.002	$1/epoch$	87.28	86.284

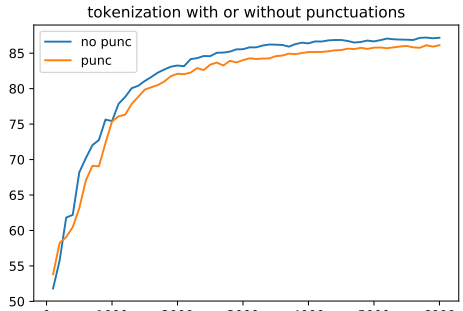

Figure 1: tokenization

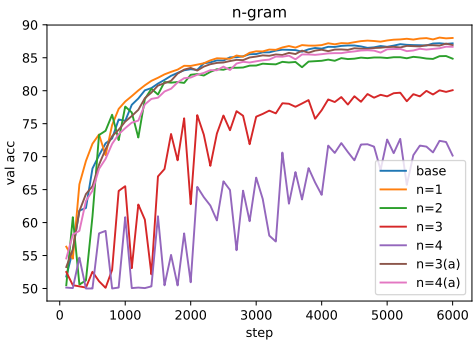

Figure 2: n-gram

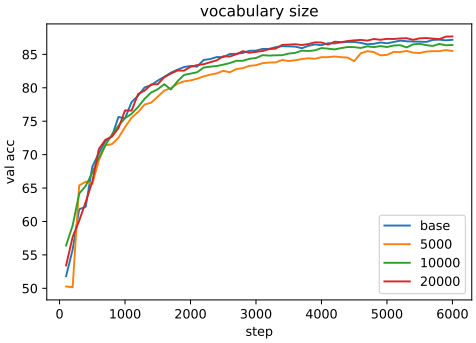

Figure 3: vocabulary size

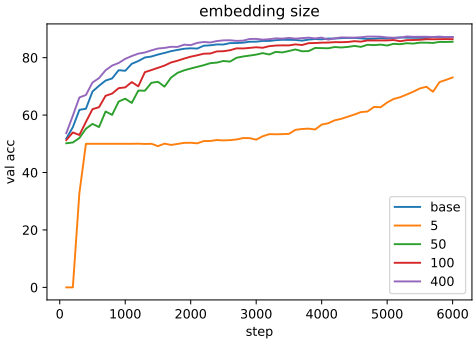

Figure 4: embedding size

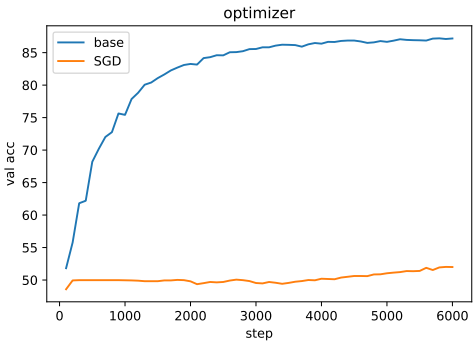

Figure 5: optimizer

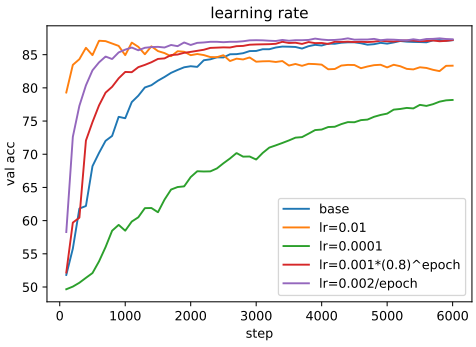

Figure 6: learning rate