1 Introduction

In Lab 1, a text classifier based on log-linear model is implemented from scratch. Logistic Regression is chosen as a representative of log-linear model for its simplicity and generality.

Note that the 3 compulsory parts (the implementation of log-linear model, the update algorithm, The results on both train and test set) in the instructions are colored redfor emphasis.

2 Environment

The program is implemented and tested on Windows 11 WSL2. The language is Python3.10.

In run.sh, ‘pip‘ is called to install the packages in requirements.txt, and then ‘python3 main.py‘ starts the program.

External data (eg. the nltk data for preprocessing) are already downloaded in local directories, in case of poor network conditon.

3 Implementation

3.1 Preprocessing

- Sample data to reduce dataset size
- Merge the Title and Content
- With the help of nltk:
  - Remove punctuations and numbers
  - Remove URLs
  - Split the Content into words
  - Filter stopwords
  - Stem and Lemmarize

3.2 Feature Extraction

Use TF-IDF as the feature.

Keep only the most (5000 in practice) frequent words for a reasonable feature size, in avoidance of excessive memory request.

Calculate TF-IDF:

$$TF(t,d) = \frac{\text{count}(t,d)}{\sum_k \text{count}(k,d)},$$

where count(t, d) means the count of term t in document d.

$$IDF(t,D) = \log \frac{N+1}{num(t,D)+1} + 1,$$

where num(t, D) means the number of documents in D that contains term t, and D is the set of all documents.

$$TF - IDF = TF \times IDF$$

Note that L2 normalization is applied to the final TF-IDF for a better performance.

3.3 Log-Linear Model

Logistic Regression Model:

$$\hat{y} = \text{softmax}\big(X_{N,F} W_{F,C} + b_C\big),$$

where N is the size of train data, F is the number of features and C is the number of classes.

Cross Entropy Loss:

$$\text{loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \log \hat{y}_{i,j},$$

where $y_{i,j} = 1$ if train text i belongs to class j, and 0 otherwise.

Gradients:

$$dW = \frac{1}{N} X^T \cdot \big(\hat{y} - y\big)$$

$$db = \frac{1}{N} \sum_{i=1}^{N} \big(\hat{y} - y\big)$$

3.4 Update Algorithm

Gradient Descend with a stepwise descending learning rate.

In practice, the learning rate is set to 0.3 for the former 8000 epochs, 0.1 for the 8000 to 9500 epochs, and 0.03 for 9500 to 10000 epochs. The rate is selected after several trials of different rates.

3.5 Evalutaion

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 Score (macro):

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = 2 \times \frac{PR}{P + R}$$

$$\text{Macro F1 Score} = \frac{1}{C} \sum_{i=1}^{C} \text{F1 Score}$$

In practice, the model reaches a satisfying evaluation results, which is completely identical to the sklearn results.

Evaluation Function	Train set	Test set
Accuracy	0.8816	0.8886
F1 Score (macro)	0.8802	0.8875

表 1: The results on both train and test set.

4 Results

4.1 Demonstration of the whole training process

```
zky@LAPTOP-2D7NGOKA:~/NLP$ python3 main.py
Preprocessing train.csv
Time consumption: 30.53 sec
<bound method NDFrame.head of         Label                                            Content
57180       4  [heavyweight, drive, new, web, servic, specamd...
44321       2  [life, ban, chanu, pratima, kumariweightlift, ...
22798       1  [israel, kill, includ, year, old, gaza, jabaly...
45766       3  [question, drug, safeti, system, emerg, arthri...
73236       3  [oct, consum, confid, sink, point, washington,...
...       ...                                                ...
107540      2  [celtic, pierc, apolog, coachboston, celtic, c...
75826       2  [sauber, ha, regret, sign, villeneuvepet, saub...
86246       4  [jon, wilcoxth, biggest, game, releas, histori...
59374       4  [aol, develop, desktop, searchaccord, sourc, f...
8565        4  [cisco, buy, p, cube, mthe, compani, sixth, bu...

[18000 rows x 2 columns]>
Preprocessing test.csv
Time consumption: 1.56 sec
<bound method NDFrame.head of       Label                                            Content
1638      3  [roger, confirm, deal, buy, amp, wireless, sta...
7198      1  [new, report, link, reput, kingpin, murderfift...
6803      1  [perfect, storm, polit, peril, unless, extend,...
431       2  [sport, graham, say, sent, syringeathen, greec...
5695      1  [american, deathsth, pentagon, ha, releas, nam...
...     ...                                                ...
6623      3  [work, dana, farber, learn, mistakesnurs, tere...
116       4  [ibm, add, midrang, server, eserv, lineupth, n...
2236      1  [walk, link, low, dementia, riskwalk, protect,...
226       1  [un, begin, second, airlift, vietnames, montag...
2897      3  [fanni, mae, crimin, probe, begunfeder, prosec...

[1140 rows x 2 columns]>
Extracting features: TFIDF, number of texts=18000
Selecting the feature words
Selected 5000 feature words: ['new', 'ha', 'hi', 'year', 'two', 'compani', 'first', 'say', 'world', 'thi'] ...
Time consumption: 42.76 sec
```

图 1: The whole training process I

```
The shape of TFIDF: (18000, 5000)
Extracting features: TFIDF, number of texts=1140
Time consumption: 2.22 sec
The shape of TFIDF: (1140, 5000)
Training the model
Iteration 0: loss=1.3862943611198906     train accuracy=0.2516111111111111
Iteration 100: loss=1.3364837548974826   train accuracy=0.8496111111111111
Iteration 200: loss=1.2896958715421207   train accuracy=0.8596111111111111
Iteration 300: loss=1.2458141738440227   train accuracy=0.8614444444444445
Iteration 400: loss=1.2047259570991269   train accuracy=0.8627777777777778
Iteration 500: loss=1.166298717109743    train accuracy=0.8639444444444444
Iteration 600: loss=1.1303863767527866   train accuracy=0.8645555555555555
Iteration 700: loss=1.0968352260704888   train accuracy=0.8658888888888889
Iteration 800: loss=1.065489068223032    train accuracy=0.867
Iteration 900: loss=1.0361933351916293   train accuracy=0.8676111111111111
Iteration 1000: loss=1.0087981481556219  train accuracy=0.8682777777777778
Iteration 1100: loss=0.9831604261370921  train accuracy=0.8692222222222222
Iteration 1200: loss=0.959145204891961   train accuracy=0.8697777777777778
Iteration 1300: loss=0.9366263393373448  train accuracy=0.8703333333333333
Iteration 1400: loss=0.9154867482671734  train accuracy=0.8708888888888889
Iteration 1500: loss=0.895618334622387   train accuracy=0.8710555555555556
Iteration 1600: loss=0.8769216869643974  train accuracy=0.8718333333333333
Iteration 1700: loss=0.8593056424860162  train accuracy=0.8723888888888889
Iteration 1800: loss=0.8426867706279811  train accuracy=0.8733888888888889
Iteration 1900: loss=0.8269888194401002  train accuracy=0.8740555555555556
Iteration 2000: loss=0.8121421538379223  train accuracy=0.8745555555555555
Iteration 2100: loss=0.7980832052129293  train accuracy=0.8749444444444444
Iteration 2200: loss=0.7847539447821797  train accuracy=0.8751111111111111
Iteration 2300: loss=0.7721013880075442  train accuracy=0.8756111111111111
Iteration 2400: loss=0.7600771338733636  train accuracy=0.8759444444444444
Iteration 2500: loss=0.7486369403867507  train accuracy=0.8763333333333333
Iteration 2600: loss=0.7377403360509065  train accuracy=0.8765555555555555
Iteration 2700: loss=0.727350260283232   train accuracy=0.8771666666666667
Iteration 2800: loss=0.7174327710868624  train accuracy=0.8774444444444445
Iteration 2900: loss=0.7079566970824714  train accuracy=0.8780555555555556
Iteration 3000: loss=0.6988934325872661  train accuracy=0.8785555555555555
Iteration 3100: loss=0.6902166722560102  train accuracy=0.8788333333333334
```

图 2: The whole training process II

```
Iteration 3200: loss=0.6819022035916821  train accuracy=0.8791111111111111
Iteration 3300: loss=0.6739277148895231  train accuracy=0.8796111111111111
Iteration 3400: loss=0.666272622286344   train accuracy=0.8803333333333333
Iteration 3500: loss=0.6589179140026791  train accuracy=0.881
Iteration 3600: loss=0.6518460100294142  train accuracy=0.8816111111111111
Iteration 3700: loss=0.6450406356712023  train accuracy=0.8818888888888889
Iteration 3800: loss=0.6384867075121986  train accuracy=0.8820555555555556
Iteration 3900: loss=0.6321702305130658  train accuracy=0.8824444444444445
Iteration 4000: loss=0.6260782050806292  train accuracy=0.8826666666666667
Iteration 4100: loss=0.6201985430726648  train accuracy=0.883
Iteration 4200: loss=0.6145199918101815  train accuracy=0.8831666666666667
Iteration 4300: loss=0.6090320652687239  train accuracy=0.8833333333333333
Iteration 4400: loss=0.6037249817093084  train accuracy=0.8838888888888888
Iteration 4500: loss=0.5985896070893817  train accuracy=0.8839444444444444
Iteration 4600: loss=0.5936174036654502  train accuracy=0.8843888888888889
Iteration 4700: loss=0.5888003832625622  train accuracy=0.8845
Iteration 4800: loss=0.584131064742383   train accuracy=0.8845555555555555
Iteration 4900: loss=0.5796024352519331  train accuracy=0.885
Iteration 5000: loss=0.5752079148797667  train accuracy=0.8856666666666667
Iteration 5100: loss=0.570941324386134   train accuracy=0.8859444444444444
Iteration 5200: loss=0.5667968557089728  train accuracy=0.8862777777777778
Iteration 5300: loss=0.5627690449789653  train accuracy=0.8866666666666667
Iteration 5400: loss=0.5588527478047832  train accuracy=0.8867222222222222
Iteration 5500: loss=0.5550431166144452  train accuracy=0.8867777777777778
Iteration 5600: loss=0.5513355798607671  train accuracy=0.8870555555555556
Iteration 5700: loss=0.547725822918522   train accuracy=0.887
Iteration 5800: loss=0.5442097705184046  train accuracy=0.8873888888888889
Iteration 5900: loss=0.5407835705784805  train accuracy=0.8877222222222222
Iteration 6000: loss=0.5374435793076925  train accuracy=0.8878888888888888
Iteration 6100: loss=0.5341863474683994  train accuracy=0.8879444444444444
Iteration 6200: loss=0.5310086076960018  train accuracy=0.8881111111111111
Iteration 6300: loss=0.5279072627836064  train accuracy=0.8882777777777778
Iteration 6400: loss=0.5248793748485535  train accuracy=0.8883888888888889
Iteration 6500: loss=0.5219221553055562  train accuracy=0.8888888888888888
Iteration 6600: loss=0.5190329555783187  train accuracy=0.8889444444444444
Iteration 6700: loss=0.5162092584878868  train accuracy=0.8891666666666667
Iteration 6800: loss=0.5134486702617078  train accuracy=0.8892777777777777
```

图 3: The whole training process III

```
Iteration 6900: loss=0.5107489131125321  train accuracy=0.8896111111111111
Iteration 7000: loss=0.5081078183409262  train accuracy=0.8897777777777778
Iteration 7100: loss=0.5055233199193331  train accuracy=0.8900555555555556
Iteration 7200: loss=0.5029934485193855  train accuracy=0.8902222222222222
Iteration 7300: loss=0.5005163259475645  train accuracy=0.8904444444444445
Iteration 7400: loss=0.49809015995736516         train accuracy=0.8904444444444445
Iteration 7500: loss=0.4957132394088976  train accuracy=0.8906666666666667
Iteration 7600: loss=0.4933839297493591  train accuracy=0.8907777777777778
Iteration 7700: loss=0.4911006687900791  train accuracy=0.891
Iteration 7800: loss=0.48886196275789773         train accuracy=0.8912222222222222
Iteration 7900: loss=0.4866663826005012  train accuracy=0.8915
Iteration 8000: loss=0.4845125605270314  train accuracy=0.8916666666666667
Iteration 8100: loss=0.4824097054768342  train accuracy=0.8916666666666667
Iteration 8200: loss=0.4803455233345809  train accuracy=0.8916666666666667
Iteration 8300: loss=0.48322293804203226         train accuracy=0.8917777777777778
Iteration 8400: loss=0.48280193956900147         train accuracy=0.8917777777777778
Iteration 8500: loss=0.48238251797039283         train accuracy=0.8918888888888888
Iteration 8600: loss=0.4819646633853024  train accuracy=0.892
Iteration 8700: loss=0.4815483660361304  train accuracy=0.892
Iteration 8800: loss=0.4811336162277046  train accuracy=0.892
Iteration 8900: loss=0.4807204043464148  train accuracy=0.892
Iteration 9000: loss=0.4803087208593569  train accuracy=0.8920555555555556
Iteration 9100: loss=0.4798985563134901  train accuracy=0.8921111111111111
Iteration 9200: loss=0.4794899013348023  train accuracy=0.8921111111111111
Iteration 9300: loss=0.4790827466274868  train accuracy=0.8921111111111111
Iteration 9400: loss=0.4786770829731298  train accuracy=0.8921666666666667
Iteration 9500: loss=0.4782729012299064  train accuracy=0.8922222222222222
Iteration 9600: loss=0.47806935062350436         train accuracy=0.8923333333333333
Iteration 9700: loss=0.47786818429477657         train accuracy=0.8924444444444445
Iteration 9800: loss=0.47766738391238284         train accuracy=0.8924444444444445
Iteration 9900: loss=0.4774669483594976  train accuracy=0.8924444444444445
Time consumption: 2520.38 sec
Evaluation of Train, size=1140, function=accuracy
Accuracy=0.8816  sklearn.metrics.accuracy_score=0.8816
Evaluation of Train, size=1140, function=macro_f1_score
F1_Score=0.8802  sklearn.metrics.f1_score=0.8802
Evaluation of Test, size=1140, function=accuracy
Accuracy=0.8886  sklearn.metrics.accuracy_score=0.8886
Evaluation of Test, size=1140, function=macro_f1_score
F1_Score=0.8875  sklearn.metrics.f1_score=0.8875
zky@LAPTOP-2D7NGOKA:~/NLP$
```

图 4: The whole training process IV

4.2 Graphics of the training process

The following are some graphics concerning the training process.

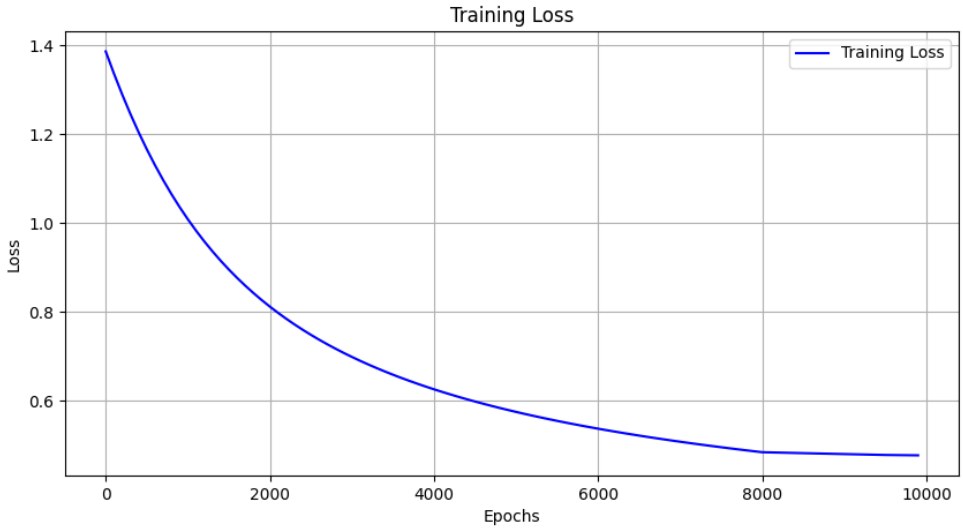

图 5: Training Loss by matplotlib.plt

图 6: Training Accuracy by matplotlib.plt

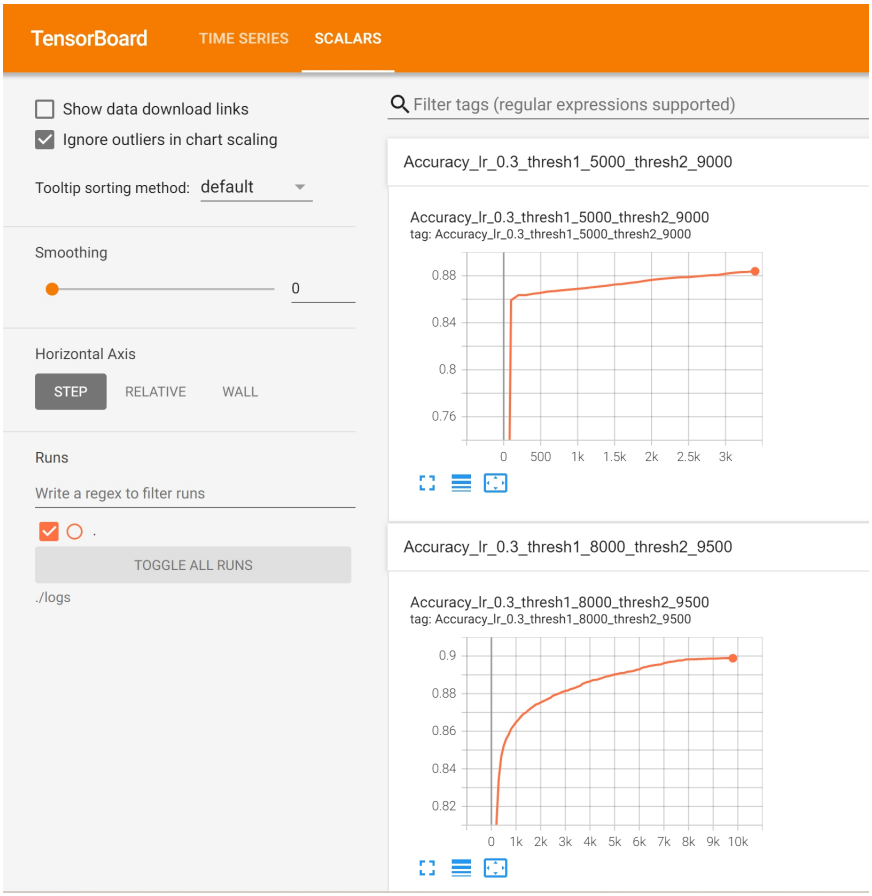

图 7: Training process by tensorboard