Text Analytics, HW 3
Louis-Charles Généreux

Github link: https://github.com/MSIA/lgo4950_msia_text_analytics_2021/tree/homework3/homework3

Question 1:

For this homework, I am using the Yelp dataset, which consists of 8.6M reviews (a numerical rating ranging from 1 to 5, along with a text review for different establishments). The most common ratings are 5/5 and 4/5 (respectively 44% and 22% of all observations). To accelerate the training of models in subsequent questions, I have created a balanced set of 500k observations (100k observations for each rating ratings); in this reduced dataset, we observe an average of 121 words per review (extrapolating this to the full dataset, we would expect over 1B words)!

Distribution of ratings

Category	Observations	Share of total
1 star	1,262,800	15%
2 star	711,378	8%
3 star	926,656	11%
4 star	1,920,037	22%
5 star	3,814,532	44%
Total	8,635,403	100%

Other data points

- Mean number of words per review: 121
- Extrapolated total words in corpus: 1.04B

Question 2:

In this question, I build a logistic regression text classifier (predicting star rating based on text reviews). I test various model parameters in distinct experiments:

1. Varying the bag-of-words representation [1gram, 1gram+2gram]
2. Varying the penalty term ['l1', 'l2']
3. Varying the regularization parameter C (inverse of regularization strength) [0.1, 1, 10]

These sets of experiments result in a total of 12 models which I compare on a set of different criteria:

* Training time (I am looking for a model that can train with no significant delays)
* Classification performance
  - Accuracy
  - Precision
  - Recall
  - F1

Given that all classes have been balanced prior to modelling (in my pre-processing step), I expect micro and macro metrics to be the same.

I observe that the best performing logistic regression model has the following parameters:

* **ngram_range**: (1,2)
* **C (regularization parameter)**: 1
* **Penalty term**: 'l1'

The resulting model results in strong out-of sample performance (over 60% accuracy), in reasonable training time (less than 2 minutes). Interestingly, precision, recall and F1 are all strongest for extreme lows and extreme high scores.

<u>Experiment results</u>

BOW representation	penalty	regularization	total time (s)	accuracy	precision	recall	F1
uni	l1	0.1	45.885	0.57043	[0.644 0.509 0.512 0.507 0.632]	[0.784 0.433 0.431 0.475 0.728]	[0.707 0.468 0.468 0.49 0.677]
uni	l2	0.1	30.707	0.5809	[0.654 0.52 0.521 0.52 0.646]	[0.785 0.451 0.448 0.487 0.733]	[0.713 0.483 0.482 0.503 0.687]
uni	l1	1	86.798	0.58694	[0.676 0.52 0.515 0.521 0.662]	[0.777 0.465 0.457 0.496 0.739]	[0.723 0.491 0.484 0.508 0.698]
uni	l2	1	55.41	0.58634	[0.677 0.52 0.512 0.52 0.662]	[0.777 0.467 0.459 0.493 0.735]	[0.723 0.492 0.484 0.507 0.697]
uni	l1	10	129.897	0.55536	[0.669 0.484 0.463 0.481 0.648]	[0.735 0.448 0.436 0.464 0.693]	[0.701 0.465 0.45 0.472 0.669]
uni	l2	10	122.789	0.57017	[0.677 0.502 0.483 0.496 0.657]	[0.756 0.458 0.447 0.477 0.712]	[0.715 0.479 0.464 0.487 0.683]
uni_bi	l1	0.1	48.063	0.56795	[0.645 0.506 0.507 0.503 0.631]	[0.779 0.43 0.435 0.473 0.722]	[0.706 0.465 0.468 0.488 0.674]
uni_bi	l2	0.1	36.717	0.5857	[0.656 0.524 0.528 0.527 0.653]	[0.782 0.46 0.46 0.497 0.73]	[0.714 0.49 0.492 0.511 0.689]
uni_bi	l1	1	89.067	0.6001	[0.692 0.536 0.528 0.535 0.675]	[0.781 0.487 0.478 0.513 0.741]	[0.734 0.51 0.502 0.524 0.706]
uni_bi	l2	1	66.878	0.59875	[0.69 0.531 0.522 0.538 0.678]	[0.779 0.483 0.48 0.511 0.74]	[0.732 0.506 0.5 0.524 0.707]
uni_bi	l1	10	275.166	0.5367	[0.665 0.462 0.437 0.464 0.638]	[0.698 0.439 0.432 0.451 0.664]	[0.681 0.45 0.434 0.457 0.651]
uni_bi	l2	10	138.023	0.56754	[0.686 0.494 0.473 0.496 0.664]	[0.736 0.464 0.46 0.477 0.701]	[0.71 0.478 0.466 0.486 0.682]

Question 3:

In this question, I build a linear SVM text classifier (predicting star rating based on text reviews). I test the same model parameters as in question 2 (bag of word representations, penalty term and regularization parameter).

I observe that the best performing SVM model has the following parameters:
* ngram_range: (1,2)
* C (regularization parameter): 0.1
* Penalty term: ‘l2’

The resulting model results in strong out-of sample performance (59% accuracy).
This is an interesting result, when compared to the one I got analyzing the logistic regression experiment. In the svm model, I select ‘l2’ penalty, with stronger regularization (lower C parameter value) as compared to ‘l1’ and a higher C for the logistic regression.

Experiment results

BOW representation	penalty	regularization	total_time (s)	accuracy	precision	recall	F1
uni	l1	0.1	77.206	0.58012	[0.648 0.525 0.527 0.514 0.635]	[0.803 0.441 0.418 0.479 0.758]	[0.717 0.479 0.466 0.496 0.691]
uni	l2	0.1	50.934	0.58224	[0.659 0.521 0.518 0.517 0.647]	[0.794 0.45 0.432 0.483 0.751]	[0.72 0.483 0.471 0.499 0.695]
uni	l1	1	119.288	0.56895	[0.66 0.501 0.495 0.497 0.645]	[0.77 0.446 0.426 0.472 0.73]	[0.711 0.472 0.458 0.484 0.684]
uni	l2	1	115.862	0.56541	[0.664 0.498 0.485 0.491 0.645]	[0.765 0.446 0.426 0.468 0.722]	[0.711 0.471 0.453 0.479 0.681]
uni	l1	10	133.917	0.53422	[0.646 0.462 0.446 0.461 0.622]	[0.714 0.428 0.41 0.446 0.673]	[0.678 0.445 0.427 0.453 0.646]
uni	l2	10	259.499	0.54045	[0.65 0.47 0.45 0.466 0.629]	[0.726 0.432 0.411 0.45 0.683]	[0.686 0.45 0.43 0.457 0.655]
uni_bi	l1	0.1	79.688	0.58775	[0.655 0.536 0.534 0.521 0.645]	[0.804 0.451 0.44 0.486 0.757]	[0.722 0.49 0.483 0.503 0.696]
uni_bi	l2	0.1	52.216	0.59345	[0.673 0.532 0.524 0.531 0.663]	[0.793 0.468 0.456 0.499 0.752]	[0.728 0.498 0.488 0.514 0.704]
uni_bi	l1	1	165.855	0.56814	[0.677 0.498 0.48 0.495 0.655]	[0.75 0.457 0.446 0.473 0.713]	[0.711 0.477 0.463 0.484 0.683]
uni_bi	l2	1	107.74	0.56207	[0.675 0.489 0.471 0.489 0.655]	[0.737 0.456 0.444 0.47 0.702]	[0.705 0.472 0.457 0.479 0.677]
uni_bi	l1	10	197.217	0.50172	[0.627 0.432 0.409 0.429 0.595]	[0.656 0.411 0.399 0.421 0.62]	[0.641 0.421 0.404 0.425 0.607]
uni_bi	l2	10	194.857	0.51775	[0.647 0.442 0.421 0.445 0.619]	[0.674 0.426 0.413 0.436 0.64]	[0.66 0.434 0.417 0.44 0.629]

Question 4:

For the purposes of this question, I have saved my top performing logistic regression model in 'pickle' format, then used it to make predictions. In the **predict.py** script, I load the model, preprocess text, then make a prediction of star rating based on the text. Below are some outputs containing:

* The text input (prior to preprocessing)
* The actual star rating
* The predicted star rating
* The predicted probability for each of the 5 classes (1 star to 5 star)

Below is a table displaying some of my results

Text	Actual	Predicted	Probability
I guess I only endured an IHOP due to the fact there are not a lot of other options and this was convenient. I can only attribute the relative busy atmosphere and people waiting for a table outside on the fact that most Americans have no idea what a really good breakfast should taste like.\nThe service was great, which is worth two stars. My server was friendly, attentive, and working hard at all of her tables. There was a nice guy making balloon creatures for the kids to keep them occupi...	2.0	2.0	[0.24590655092834965, 0.2690908206460493, 0.25005195496704, 0.1431120985298537, 0.09183857492870738]
Upon entering the establishment, i was greeted by a waitress who led me to my group. The owner had walked past me mumbling away in in fury. \n\nAs we ordered our food, i decided to go for the calamari combo. A friend who sat next to me ordered his meal. When we got our food, the waitress forgot his appetizer order and told him that she never got his order. The food itself was good. Calamari was cooked just right and the rice was nice and flavourful. \n\nI had to leave earlier so i figured i ...	1.0	1.0	[0.3749014914915614, 0.13241710352479188, 0.2442494679854127, 0.13566386723188514, 0.11276806976634879]
Very rude staff. I immediately found a piece of hair in my food and the cook got made that the waitress brought it back. I will never go back.	1.0	1.0	[0.3319493480228793, 0.23100660548255394, 0.1692538795601774, 0.0877752089315689, 0.18001495800282036]

Outputs from key .py files

dataset_stats.py

```
(base) [20:44:36] louisgenereux:HW3 git:(main) $ python dataset_stats.py
- JSON format review data has been read
- All reviews categorized based on their ratings
- Distribution of ratings:
   stars  observations  pct share of total
0      1       1262800           14.623521
1      2        711378            8.237925
2      3        926656           10.730895
3      4       1920037           22.234481
4      5       3814532           44.173179
5  total       8635403          100.000000
- Balanced subset of reviews created, with 100000 observations per class
- Description of word counts:
          word_count
count  500000.000000
mean      121.029352
std       109.807441
min         0.000000
25%        49.000000
50%        88.000000
75%       156.000000
max      1044.000000
- Balanced data set saved as csv
```

logistic_regression.py

```
(base) [14:56:55] louisgenereux:HW3 git:(main) $ python logistic_regression.py
- Balanced data loaded with (500000 rows)
- Text has been preprocessed
- Training set (400000 rows) and Test set (100000 rows) split
- 1gram bag-of-word representations created
- 1gram+2gram bag-of-word representations created
- Ready for LOGISTIC REGRESSION
Time:  45.885  Acc:  0.57043
Time:  30.707  Acc:  0.5809
Time:  86.798  Acc:  0.58694
Time:  55.41  Acc:  0.58634
Time:  129.897  Acc:  0.55536
Time:  122.789  Acc:  0.57017
Time:  48.063  Acc:  0.56795
Time:  36.717  Acc:  0.5857
Time:  89.067  Acc:  0.6001
Time:  66.878  Acc:  0.59875
Time:  275.166  Acc:  0.5367
Time:  138.023  Acc:  0.56754
```

```
- Logistic regression results:
   BOW representation penalty  regularization  ...
0                 uni       l1             0.1  ...
1                 uni       l2             0.1  ...
2                 uni       l1             1.0  ...
3                 uni       l2             1.0  ...
4                 uni       l1            10.0  ...
5                 uni       l2            10.0  ...
6              uni_bi       l1             0.1  ...
7              uni_bi       l2             0.1  ...
8              uni_bi       l1             1.0  ...
9              uni_bi       l2             1.0  ...
10             uni_bi       l1            10.0  ...
11             uni_bi       l2            10.0  ...

[12 rows x 8 columns]
- Logistic regression results stored in csv
- Best logit model saved in Pickle
```

svm.py

```
(base) [16:00:25] louisgenereux:HW3 git:(main) $ python svm.py
- Balanced data loaded with (500000 rows)
- Text has been preprocessed
- Training set (400000 rows) and Test set (100000 rows) split
- 1gram bag-of-word representations created
- 1gram+2gram bag-of-word representations created
- Ready for SVM
Time:  77.206  Acc:  0.58012
Time:  50.934  Acc:  0.58224
Time:  119.288  Acc:  0.56895
Time:  115.862  Acc:  0.56541
Time:  133.917  Acc:  0.53422
Time:  259.499  Acc:  0.54045
Time:  79.688  Acc:  0.58775
Time:  52.216  Acc:  0.59345
Time:  165.855  Acc:  0.56814
Time:  107.74  Acc:  0.56207
Time:  197.217  Acc:  0.50172
Time:  194.857  Acc:  0.51775
- Logistic regression results:
```

```
- Logistic regression results:
   BOW representation penalty  regularization
0                 uni       l1             0.1
1                 uni       l2             0.1
2                 uni       l1             1.0
3                 uni       l2             1.0
4                 uni       l1            10.0
5                 uni       l2            10.0
6              uni_bi       l1             0.1
7              uni_bi       l2             0.1
8              uni_bi       l1             1.0
9              uni_bi       l2             1.0
10             uni_bi       l1            10.0
11             uni_bi       l2            10.0

[12 rows x 9 columns]
- Best svm model saved in Pickle
```

predict.py

```
(base) [17:29:27] louisgenereux:HW3 git:(main) $ python predict.py
- Balanced data loaded with (500000 rows)
- Text has been preprocessed
- Model loaded from Pickle
- BOW representations created
- Forecasts created
- Forecasts saved to JSON format
```