HW-3

Name: Ren, Jing

NetID: jrb1992

Link to the git repository branch:

https://github.com/MSIA/jrb1992_msia_text_analytics_2020/tree/homework3

Problem 1

The text corpus ('yelp_academic_dataset_review.json') is chosen from the Yelp reviews dataset (https://www.yelp.com/dataset). Due to limited computing resources, I'm using only the first 500,000 training examples from this large dataset. Each review would include the following components: review_id, user_id, busines_id, stars, useful, funny, cool, text, date.

Dataset Summary Statistics:

1) Number of documents: 500,000
2) Number of labels: 5
3) Average / mean word length of documents: 123.175352
4) Label distribution:

Label	Count (Number of Reviews)
1	70468
2	40577
3	55778
4	112802
5	220375

As for label distribution, 5-star ratings are the most frequent values, while 2-star ratings being the least frequent ones.

- Source Code:

  https://github.com/MSIA/jrb1992_msia_text_analytics_2020/blob/homework3/dataset_stats.py

Problem 2

To build a logistic regression text classifier, I chose the scikit-learn library (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html). I also created a binary label that classifies stars > 3 as "1" (good ratings) and stars <= 3 as "0" (bad ratings).

I tested out different sets of bag-of-word representation, inverse of regularization strength (C), and maximum number of iterations taken for the solvers to converge. There were 5 evaluation metrics that I calculated to compare results: precision, recall, F1-score, accuracy, micro-averaged F1-score.

As we can see from the table below, 1-gram+2gram models have better performance in terms of predictive power, comparing to 1-gram models. As we decrease the inverse of regularization strength (C), smaller values specify stronger regularization. The models with C = 0.5 have slightly better performance comparing to those with C = 5. Even though running time changes a bit, maximum number of iterations does not have much of impact on results in this case.

	1-gram, C=0.5, max_iter=400	**1-gram, C=5, max_iter=400**	**1-gram+2-gram, C=0.5, max_iter=400**	**1-gram+2-gram, C=5, max_iter=100**
Precision	0.761189680628	0.761654487905	0.827599852039	0.827843589655
Recall	0.712204520462	0.708569247407	0.874337698690	0.873910554106
F1-score	0.735882808648	0.734153495576	0.850327028460	0.850253549023
Accuracy	0.659072727273	0.657787878787	0.79473939393	0.794721212121
Micro-averaged F1-score	0.659072727273	0.657787878787	0.79473939393	0.794721212121

Out of these 4 models, 1-gram+2gram model with C = 0.5 and max_iter = 400 has the best performance.

- Source Code: https://github.com/MSIA/jrb1992_msia_text_analytics_2020/blob/homework3/logistic_regression.py

Problem 3

To build a linear SVM text classifier, I chose the scikit-learn library (https://scikit-learn.org/stable/modules/svm.html).

I tested out different sets of bag-of-word representation, inverse of regularization strength (C), and loss function. There were 5 evaluation metrics that I calculated to compare results: precision, recall, F1-score, accuracy, micro-averaged F1-score.

As we can see from the table below, 1-gram+2gram models have better performance in terms of predictive power, comparing to 1-gram models. The models with C = 0.5 have similar performance comparing to those with C = 5, though higher C value could end up with a smaller margin. By specifying loss function to hinge or squared hinge, I want to find out the different loss functions' impact on adjusting weights for iterations, which is not quite obvious in this case.

	1-gram, C=0.5, loss=hinge	**1-gram, C=0.5, loss=squared_hinge**	**1-gram+2-gram, C=0.5, loss=hinge**	**1-gram+2-gram, C=5, loss=squared_hinge**
Precision	0.7600279552	0.76171409597843	0.8282997088	0.826364657367478
Recall	0.7017167576	0.70886006925195	0.8738651131	0.875991747930166
F1-score	0.7297092957	0.73433726715968	0.8504725387	0.850454838225822
Accuracy	0.6533333333	0.65796969696969	0.7950848484	0.794557575757575
Micro-averaged F1-score	0.6533333333	0.65796969696969	0.7950848484	0.794557575757575

Out of these 4 models, 1-gram+2gram model with C = 0.5 and loss = hinge has the best performance. Thus, this model will be used in Problem 4 to write prediction script.

- Source Code:
  https://github.com/MSIA/jrb1992_msia_text_analytics_2020/blob/homework3/svm.py

Problem 4

Loading in the best SVM model and TfidfVectorizer (saved from svm.py), the model predicts the label for next 50,000 lines of Yelp reviews from row 600,000 to 650,000. Note that I created a binary label earlier: "1" stands for good ratings that have more than 3 stars, "0" stands for bad ratings that has less or equal to 3 stars.

The confidence score is generated by calling decision_function for SVM, which computes the signed distance of a point from the boundary (class 0: negative value, class 1: positive value). A value that is closer to 0 means that the point is closer to boundary.

Here are several runs of my script producing different label predictions:

Review	Actual Label	Predicted Label	Confidence
The food was Absolutely delicious and the service was great!! I enjoyed the ambiance on date night with bae.	1	1	0.906782218416399
What can I say. I have gave them multiple chances. Everytime my order is messed up. I am a vegetarian so why do they always insist on giving me a burrito with meat in it.	0	0	0.987302477312259
This place earns 3 stars simply for it's Lunch Specials. How can you argue with $1 Thai Iced Tea?? You won't be able to find that deal anywhere else	0	1	2.01157140671526
Last few times coming here for happy hour has been a total let down. Not sure what has happened, but quality control seems to be lacking now. Sushi rolls were unevenly sized	0	0	-0.21601037534741
This place is solid Italian food. The owner is very friendly, and he told me they run the whole kitchen on 4 burners. Try the lasagna it's the best I'm town	1	1	1.37885838495372

- Source Code: https://github.com/MSIA/jrb1992_msia_text_analytics_2020/blob/homework3/predict_svm.py