CSE 5243 HW3

Name: Neng Shi

Dataset Split

I randomly split the dataset into training, validation, and testing sets by generating a randomly permuted array ranging from 0 to 2999. The first 70% of indices are for the training set, the next 15% are for the validation set, and the rest are for the testing set.

Feature Selection

original feature vector (FV-1)

The original feature vector is unigram bag-of-words features. I first transform all the words into lowercase, then discard stop words. I count all occurrences of each word. Finally, I normalize the occurrences by dividing the number of words in the sentence.

pared down feature vector (FV-2)

For the pared down feature vector, on top of the original feature vector, I only keep words that appear at least twice.

Classification Algorithms

The first classifier (CL-1)

The first classifier I use is K-nearest neighbor, where I use cosine distance as the distance metric.

The second classifier (CL-2)

The second classifier I use is the decision tree, which is implemented with the help of

package sklearn.

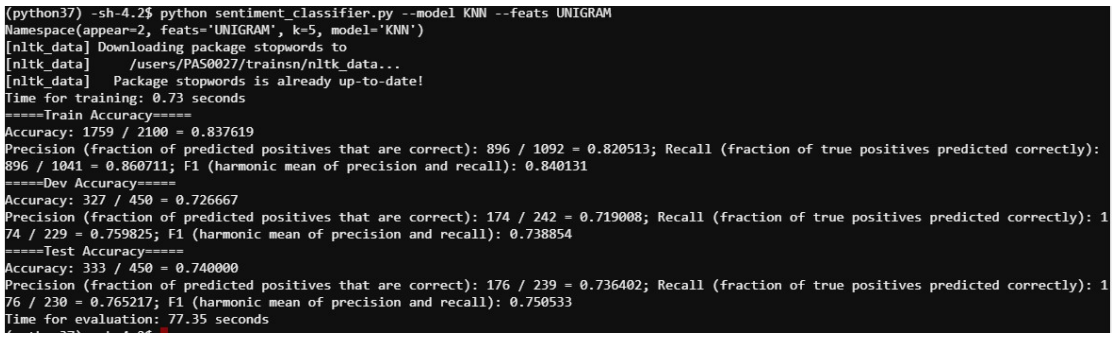
The third classifier (CL-3)

The third classifier I use is the logistic regression, which was implemented during taking CSE 5525, offered by Dr. Huan Sun.

Results

FV-1; CL-1

(1) Time to build the classifier model is 0.73s
(2) Time to classify a new tuple is 77.35 / 3000 = 0.026s
(3) After the hyperparameter tuning on the validation set, I select the hyperparameter k=5 and report the accuracy. The accuracy of the classifier on training, validation, and testing sets are 0.8376, 0.7267, and 0.7400, respectively.
(4) As a lazy learner, the advantage of KNN is quick training. However, the disadvantage is that it takes a long time to predict since it needs to scan all the candidates and get the k nearest neighbors.
The discussion of feature selection will be in the next subsection.



FV-2; CL-1

(1) Time to build the classifier model is 0.64s
(2) Time to classify a new tuple is 64.38 / 3000 = 0.021s
(3) After the hyperparameter tuning on the validation set, I select the hyperparameter k=9 and report the accuracy. The accuracy of the classifier on training, validation, and testing sets are 0.8195, 0.7400, and 0.7422, respectively.
(4) After using the pared down feature vector, we only focus on frequent words. The length of the feature reduces from 4168 to 1608. It turns out that using frequent words only can achieve as good performance as that using original features. Moreover, it speeds up the

process of classifying a new tuple.

```
(python37) -sh-4.2$ python sentiment_classifier.py --model KNN --k 9 --feats IMPORTANT --appear 2
Namespace(appear=2, feats='IMPORTANT', k=9, model='KNN')
[nltk_data] Downloading package stopwords to
[nltk_data]     /users/PAS0027/trainsn/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
corpus length is 1608
Time for training: 0.64 seconds
=====Train Accuracy=====
Accuracy: 1721 / 2100 = 0.819524
Precision (fraction of predicted positives that are correct): 883 / 1104 = 0.799819; Recall (fraction of true positives predicted correctly):
883 / 1041 = 0.848223; F1 (harmonic mean of precision and recall): 0.823310
=====Dev Accuracy=====
Accuracy: 333 / 450 = 0.740000
Precision (fraction of predicted positives that are correct): 178 / 244 = 0.729508; Recall (fraction of true positives predicted correctly): 1
78 / 229 = 0.777293; F1 (harmonic mean of precision and recall): 0.752643
=====Test Accuracy=====
Accuracy: 334 / 450 = 0.742222
Precision (fraction of predicted positives that are correct): 182 / 250 = 0.728000; Recall (fraction of true positives predicted correctly): 1
82 / 230 = 0.791304; F1 (harmonic mean of precision and recall): 0.758333
Time for evaluation: 64.38 seconds
```

FV-1; CL-2

(1) Time to build the classifier model is 4.57s
(2) Time to classify a new tuple is 1.24 / 3000 = 0.0004s
(3) The accuracy of the classifier on training, validation, and testing sets are 0.9981, 0.7156, and 0.7267 respectively.
(4) For the decision tree algorithm, we see a high training accuracy but do not see increased validation and testing accuracy compared to KNN, which indicates that there might be an overfitting problem of the classifier. The reason can be that we do not use pruning for the decision tree. In the feature, I will consider pruning the decision tree.
As an eager learner, the advantage of the decision tree is quick predicting. Of course, it takes longer to train compared to the KNN classifier.
The discussion of feature selection will be in the next subsection.

```
(python37) -sh-4.2$ python sentiment_classifier.py --model DT --feats UNIGRAM
Namespace(appear=2, feats='UNIGRAM', k=5, model='DT')
[nltk_data] Downloading package stopwords to
[nltk_data]     /users/PAS0027/trainsn/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Time for training: 4.57 seconds
=====Train Accuracy=====
Accuracy: 2096 / 2100 = 0.998095
Precision (fraction of predicted positives that are correct): 1038 / 1039 = 0.999038; Recall (fraction of true positives predicted correctly):
 1038 / 1041 = 0.997118; F1 (harmonic mean of precision and recall): 0.998077
=====Dev Accuracy=====
Accuracy: 322 / 450 = 0.715556
Precision (fraction of predicted positives that are correct): 159 / 217 = 0.732719; Recall (fraction of true positives predicted correctly): 1
59 / 229 = 0.694323; F1 (harmonic mean of precision and recall): 0.713004
=====Test Accuracy=====
Accuracy: 327 / 450 = 0.726667
Precision (fraction of predicted positives that are correct): 159 / 211 = 0.753555; Recall (fraction of true positives predicted correctly): 1
59 / 230 = 0.691304; F1 (harmonic mean of precision and recall): 0.721088
Time for evaluation: 1.24 seconds
```

FV-2; CL-2

(1) Time to build the classifier model is 1.91s
(2) Time to classify a new tuple is 0.90 / 3000 = 0.0003s
(3) The accuracy of the classifier on training, validation, and testing sets are 0.9962, 0.7200, and 0.7356 respectively.
(4) After using the pared down feature vector, we only focus on frequent words. The length

of the feature reduces from 4168 to 1608. For the decision tree, using frequent words only does not solve the overfitting problem. As I mentioned before, I plan to prune the decision tree in the future.

From the efficiency costs point of view, using frequent words only speeds up the training and predicting process.

```
(python37) -sh-4.2$ python sentiment_classifier.py --model DT --feats IMPORTANT --appear 2
Namespace(appear=2, feats='IMPORTANT', k=5, model='DT')
[nltk_data] Downloading package stopwords to
[nltk_data]     /users/PAS0027/trainsn/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
corpus length is 1608
Time for training: 1.93 seconds
=====Train Accuracy=====
Accuracy: 2092 / 2100 = 0.996190
Precision (fraction of predicted positives that are correct): 1035 / 1037 = 0.998071; Recall (fraction of true positives predicted correctly):
 1035 / 1041 = 0.994236; F1 (harmonic mean of precision and recall): 0.996150
=====Dev Accuracy=====
Accuracy: 324 / 450 = 0.720000
Precision (fraction of predicted positives that are correct): 162 / 221 = 0.733032; Recall (fraction of true positives predicted correctly): 1
62 / 229 = 0.707424; F1 (harmonic mean of precision and recall): 0.720000
=====Test Accuracy=====
Accuracy: 331 / 450 = 0.735556
Precision (fraction of predicted positives that are correct): 158 / 205 = 0.770732; Recall (fraction of true positives predicted correctly): 1
58 / 230 = 0.686957; F1 (harmonic mean of precision and recall): 0.726437
Time for evaluation: 0.90 seconds
```

FV-2; CL-3

(1) Time to build the classifier model is 1.38s
(2) Time to classify a new tuple is 0.75 / 3000 = 0.00025s
(3) The accuracy of the classifier on training, validation, and testing sets are 0.9176, 0.7622, and 0.7711 respectively.
(4) The overfitting problem is relieved due to early stopping, and I get the best performance on the testing set.

Meanwhile, as an eager learner, the advantage of logistic regression is the same as the decision tree, which is quick predicting. However, it takes longer to train compared to the KNN classifier.

```
(python37) -sh-4.2$ python sentiment_classifier.py --model LR --feats IMPORTANT --appear 2
Namespace(appear=2, feats='IMPORTANT', k=5, model='LR')
[nltk_data] Downloading package stopwords to
[nltk_data]     /users/PAS0027/trainsn/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
corpus length is 1608
epoch 0, loss: 0.598242, accuracy: 0.752381
epoch 1, loss: 0.488076, accuracy: 0.847143
epoch 2, loss: 0.436175, accuracy: 0.870000
epoch 3, loss: 0.401338, accuracy: 0.879524
epoch 4, loss: 0.375710, accuracy: 0.890952
epoch 5, loss: 0.354951, accuracy: 0.905238
epoch 6, loss: 0.337692, accuracy: 0.901905
epoch 7, loss: 0.323084, accuracy: 0.910476
training loss: 0.628986
Time for training: 1.38 seconds
=====Train Accuracy=====
Accuracy: 1927 / 2100 = 0.917619
Precision (fraction of predicted positives that are correct): 944 / 1020 = 0.925490; Recall (fraction of true positives predicted correctly):
944 / 1041 = 0.906820; F1 (harmonic mean of precision and recall): 0.916060
=====Dev Accuracy=====
Accuracy: 343 / 450 = 0.762222
Precision (fraction of predicted positives that are correct): 169 / 216 = 0.782407; Recall (fraction of true positives predicted correctly): 1
69 / 229 = 0.737991; F1 (harmonic mean of precision and recall): 0.759551
=====Test Accuracy=====
Accuracy: 347 / 450 = 0.771111
Precision (fraction of predicted positives that are correct): 164 / 201 = 0.815920; Recall (fraction of true positives predicted correctly): 1
64 / 230 = 0.713043; F1 (harmonic mean of precision and recall): 0.761021
Time for evaluation: 0.75 seconds
```

Acknowledgments

The framework of the code is from homework one of CSE 5525.