Eric Burton Martin
09/21/2023

PA1 – Logistic Regression Classification

• What additional features you included/tried in your classifier

1. I went through the negative and positive reviews and found the count of each word appearance in order to generate a list of the most common words in the negative reviews. I realized the most common words were punctuation, proper nouns, and movie specific words.
    a. In order to address this, I only added alpha characters to the word list (I now realize that I should have kept exclamation points, but oh well). I then manually, painstakingly, chose the words in the top N positive sentiment that were not proper nouns or movie specific terms and repeated the process for the negative sentiment words.
    b. To see the default bag of words you can uncomment the print methods in the main function.

2. I also generated the most common bigrams for the negative and positive reviews. In order to filter out the non-sentiment-based bigrams, did the following:
    a. I created a list that was the union of the positive and negative common sentiment words
    b. I removed the bigrams that did not include at least one of the words within the union of common pos and neg words. This should help remove bigrams that are not indicative of sentiment.
    c. To see the original bigrams and the revised bigrams, you can uncomment the print methods in the main function.

3. I then created a list of common conclusive words, such as “conclusion”, “overall”, “rating”, etc. The thought process of generating this list was originally to check if the last two sentences of each doc were reached, and if so, check if any conclusive words were hit. If they were, then I would weigh the sentiment hits higher since they would most likely be indicative of a final sentiment.

    a. I didn't want to add any return functions to the featurize() function so instead of checking for the last two sentences, I just checked for any conclusive words and then set a flag to true which probably not as good, but oh well.
    b. If a single word sentiment hit occurred with the conclusive word flag triggered, the resulting hit will have a higher weight.
    c. If a bigram hit occurred with the conclusive word flag triggered, the resulting bigram will have a much higher weight.

• How you tuned the hyperparameters, and their final values

I used a grid search to try the following combinations:

```
batch_sizes = [1, 2, 3, 10]
n_epochs_list = [1, 2, 5, 10]
etas = [0.01, 0.05, 0.1, 0.15, 0.2]
```

I tried this grid search using various combination of my features. Thankfully the best result was using all my chosen features, but only just barely. I am determining the ‘best’ result to be the result with the highest accuracy.

From my search, the best hyperparameters were:

batch_size=1, n_epochs=1, eta=0.1

My thoughts on why this is the case are as follows:

1. Batch size:
    a. It was interesting but utilizing a batch size of 1 increased the precision of the classification model. Looking into this I came to the conclusion that smaller batch sizes are usually noisier which can act as a kind of regularization. This can end up helping the model generalize better to unseen data. Larger batch sizes produce less noisy gradients but might result in overfitting or less generalizable models.
    b. Larger batch sizes seemed to increase recall but reduce precision. This seems to be due to the smoother gradient updates that can help the model learn overall patterns better which can reducing FNs and increase TPs.
2. Epochs:
    a. Adding more epochs may be leading to overfitting to the training data in the logistic regression model and reducing the overall performance.
3. Learning Rate:
    a. Changing the learning rate from 0.01 to 0.1 had shown improvement in overall classification performance. I presume that using a lower learning rate and only one epoch acts as a form of regularization which makes the model work better on the unseen test data.

• Your evaluation results on the development set

It seems that I encountered a trade-off in machine learning. Improving recall at the expense of precision and accuracy is a classic problem. When I reduced the learning rate, the precision and accuracy went down but the recall kept getting higher. Because of this, I ended up just using the overall accuracy as my metric to go off.

My best result was using all of my features (the cleaned sets of neg and pos words, cleaned set of neg and pos bigrams, and the conclusive words weighting) led to the best results which are seen below. What is interesting though is that I was able to get comparable results using less features using the grid search to find the best hyperparameters for the features. Using only my cleaned bag of words lists was pretty much just as good as using all of my features.

Threats to validity. When I did these alternate feature tests I did not change the overall number of features that the logistic regression was using (i.e. the vector was the same size for all attempts). This may have an effect since the vector will have all zeros for unused features.

See a few evaluation results below:

Eric Burton Martin
09/21/2023

The overall <u>best grid search results</u>:

Features used:

1. cleaned bag of words
2. cleaned bigrams
3. conclusive words

<u>Hyperparameters: batch_size=1, n_epochs=1, eta=0.1</u>

Average Train Loss: nan

Metrics for Positive Class:
Precision: 0.6746
Recall: 0.8586
F1: 0.7556
Metrics for Negative Class:
Precision: 0.8108
Recall: 0.5941
F1: 0.6857
Overall Accuracy of Model: **0.7556**

The best grid search using the following features:

1. cleaned bag of words
2. cleaned bigrams

<u>Hyperparameters: batch_size=3, n_epochs=5, eta=0.2</u>

Metrics for Positive Class:
Precision: 0.6667
Recall: 0.8081
F1: 0.7306
Metrics for Negative Class:
Precision: 0.7625
Recall: 0.604
F1: 0.674
Overall Accuracy of Model: **0.7306**

The best grid search using the following features

1. cleaned bag of words

<u>Hyperparameters: batch_size: 1, n_epochs: 2, eta: 0.2</u>

Metrics for Positive Class:
Precision: 0.6471
Recall: 0.8889
F1: 0.7489
Metrics for Negative Class:
Precision: 0.8281
Recall: 0.5248
F1: 0.6424
Overall Accuracy of Model: **0.7489**

The best grid search using the following features:

1. cleaned bag of words
2. conclusive words

<u>Hyperparameters: batch_size: 1, n_epochs: 1, eta: 0.15</u>

Average Train Loss: nan

Metrics for Positive Class:
Precision: 0.6268
Recall: 0.899
F1: 0.7386
Metrics for Negative Class:
Precision: 0.8276
Recall: 0.4752
F1: 0.6038

Overall Accuracy of Model: **0.7386**

The best grid search using the following features:

1. cleaned bigrams
2. conclusive words

<u>Hyperparameters: batch_size: 1, n_epochs: 1, eta: 0.2</u>

Metrics for Positive Class:
Precision: 0.5052
Recall: 0.9798

Eric Burton Martin
09/21/2023

F1: 0.6667

Overall Accuracy of Model: **0.6667**

Metrics for Negative Class:

Precision: 0.75

Recall: 0.0594

F1: 0.1101