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.

RESULTS

The conclusive word feature as implemented had barely any effect on the evaluation and in some cases, had no noticeable effect on the results. This was disappointing as I thought it was an innovative little idea and coming up with the ‘conclusive words’ took some time. But, in my best test, it increased the precision by 1 %. If implemented the way I had originally hoped (only looking for conclusive words in the last two sentences), maybe it could provide more use. The bag of words, when tested alone did great. The bigrams tested alone did quite bad but this was due to how I ‘cleaned’ the bigrams based on the bag of words results.

Eric Burton Martin
09/21/2023

• How you tuned the hyperparameters, and their final values

1. Changed batch size: Used batch_size=1
    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. Since I didn't want to mess with the loss function I used the batch size of 1 as a form of regularization.
    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. Changing epochs: Used n_epochs=1
    a. Adding more epochs simply overfit the data and reduced the overall performance so I decided to keep this at 1.
3. Learning Rate: Used eta=0.01
    a. Changing the learning rate from 0.1 to 0.01 had shown improvement in overall classification performance. In my best scenario, my average loss when printed is 'nan'. I found out I could fix that with lowering the learning rate and adding epochs but I didn't like my results as much as with 0.01.

• 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 attempted to fix the 'nan' value for my average train loss by reducing the learning rate, the precision and accuracy went down but the recall kept getting higher. Even though these are all pretty similar, I think that the use of the bag of words, bigrams, and conclusive words led to the best results. It is funny that I would only choose 1 epoch since I don't think I have ever just used 1 epoch before. But, I am satisfied enough with the results. Fully reading the instructions, I used some complicated lexicon (the opinion lexicon by Minqing Hu and Bing Liu) and when I ran it I got around .61 accuracy or something like that. Then I read the instructions further and stopped using it but it is fun to see that I got better than that dataset (although I didn't play with hyperparameters with that dataset).

words + bigram + conclusive words

Average Train Loss: nan
Precision: 0.71
Recall: 0.73
F1: 0.72
Accuracy: 0.71

bigrams + no words + conclusive words

Average Train Loss: 0.649795892498662
Precision: 0.63
Recall: 0.17
F1: 0.27
Accuracy: 0.54

words + bigram + no conclusive words

Average Train Loss: 0.47955403043614564
Precision: 0.7
Recall: 0.74
F1: 0.72
Accuracy: 0.71

words + no conclusive words + no bigrams

Average Train Loss: 0.5990533887000353
Precision: 0.73
Recall: 0.7
F1: 0.71
Accuracy: 0.72