Adrien Protzel

AI 534 Machine Learning HW2

0. Sentiment Classification Task and Dataset
    1. Question: why is each of these steps necessary or helpful for machine learning?
        a. Cleaning data helps normalize and standardize it, allowing perceptrons to focus on meaningful patterns like word usage and sentence structure. Proper data cleaning ensures that all input is usable and consistent, which improves the model's ability to learn effectively and make accurate predictions.

1. Naive Perceptron Baseline
    1. Take a look at svector.py, and briefly explain why it can support addition, subtraction, scalar product, dot product, and negation.
        a. The class holds multiple functions for adding and subtracting by taking in self and other and doing math of the vectors.

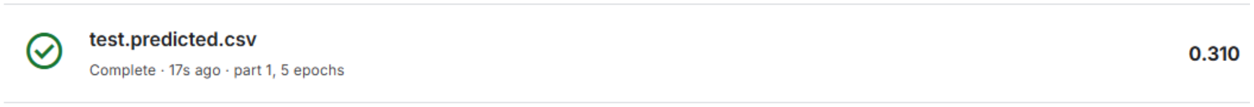
    2. Take a look at train.py, and briefly explain train() and test() functions.
        a. Test() is getting an arrow rate of the model by reading in devfile and then returning the vector of labels divided by the index. Train() is training a perceptron on train and then testing on dev. It uses test() in its print display for checking error rate.

    3. There is one thing missing in my train.py: the bias dimension! Try adding it. How did you do it? Did it improve error on dev?
        a. best dev err 30.1%, |w|=16743, time: 2.1 secs => best dev err 28.9%, |w|=16743, time: 2.0 secs. Yes the error rate improved with the addition of

```python
def make_vector(words):
    v = svector()
    for word in words:
        v[word] += 1
    v["<bias>"] = 1 # added bias
    return v
```

In the make_vector function after the words vector was created.

    4. Using your best model (in terms of dev error rate), predict the semi-blind test data, and submit it to Kaggle (follow instructions from Part 5). What are your error rate and ranking on the public leaderboard? Take a screenshot.
        a. Just as you predicted, my public score is ~31%, my dev error rate for test in my program was 37.7%


test.predicted.csv
Complete · 17s ago · part 1, 5 epochs
0.310

5. Wait a second, I thought the data set is already balanced (50% positive, 50% negative). I remember the bias being important in highly unbalanced data sets. Why do I still need to add the bias dimension here??
    a. To shift the decision boundary so it better separates the positive and negative points, even in a balanced dataset. The bias term allows the model to adjust the boundary's position without relying solely on the input features.

2. Average Perceptron

1. Train for 10 epochs and report the results. Did averaging improve the dev error rate? Did it also make dev error rates more stable?
    a. best dev err 26.3%, |w|=16744, time: 2.2 secs.
    b. The average improved slightly by about 2% and the error rates were pretty stable around 26-27% for most of them.

2. Did smart averaging slow down training?
    a. It is just as if not faster than the normal perceptron.

3. What are the top 20 most positive and top 20 most negative features? Do they make sense?

```
Top 20 most negative features:
boring: -14.9133
generic: -13.0597
dull: -12.8838
badly: -11.8677
routine: -11.7137
fails: -11.1441
ill: -11.0039
too: -10.5788
instead: -10.2225
tv: -10.2003
attempts: -9.9460
unless: -9.9310
incoherent: -9.8561
neither: -9.8469
flat: -9.7828
seagal: -9.6624
problem: -9.6312
scattered: -9.5949
worst: -9.5857
suffers: -9.5715

Top 20 most positive features:
```

```
engrossing: 12.1910
triumph: 11.3317
unexpected: 11.1280
rare: 11.1244
provides: 10.9798
french: 10.8039
skin: 10.6206
treat: 10.5877
pulls: 10.4002
culture: 10.2417
cinema: 10.2386
dots: 10.2072
wonderful: 10.1635
refreshingly: 10.0542
open: 9.8958
powerful: 9.7531
delightful: 9.7266
imax: 9.6073
smarter: 9.3821
flaws: 9.3799
```

a. I would say the positive and negative words are about 50/50 in terms of making sense. Things like, “seagull, too, imax, dots, flaws” don't seem to make much sense by themselves, perhaps in sentence context they are more seen.

4. Show 5 negative examples in dev where your model most strongly believes to be positive. Show 5 positive examples in dev where your model most strongly believes to be negative. What observations do you get?

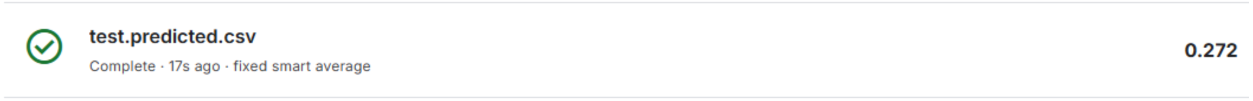
```
Top 5 negative examples misclassified as positive:
35.1060: ` in this poor remake of such a well loved classic , parker
exposes the limitations of his skill and the basic flaws in his vision '
30.0698: how much you are moved by the emotional tumult of fran ois and
mich le 's relationship depends a lot on how interesting and likable you
find them
29.7196: bravo reveals the true intent of her film by carefully selecting
interview subjects who will construct a portrait of castro so
predominantly charitable it can only be seen as propaganda
28.7119: mr wollter and ms seldhal give strong and convincing performances
, but neither reaches into the deepest recesses of the character to
unearth the quaking essence of passion , grief and fear
```

```
23.7235: an atonal estrogen opera that demonizes feminism while gifting
the most sympathetic male of the piece with a nice vomit bath at his
wedding

Top 5 positive examples misclassified as negative:
-43.6059: the thing about guys like evans is this you 're never quite sure
where self promotion ends and the truth begins but as you watch the movie
, you 're too interested to care
-40.5688: neither the funniest film that eddie murphy nor robert de niro
has ever made , showtime is nevertheless efficiently amusing for a good
while before it collapses into exactly the kind of buddy cop comedy it set
out to lampoon , anyway
-29.0424: even before it builds up to its insanely staged ballroom scene ,
in which 3000 actors appear in full regalia , it 's waltzed itself into
the art film pantheon
-27.2231: if i have to choose between gorgeous animation and a lame story
( like , say , treasure planet ) or so so animation and an exciting ,
clever story with a batch of appealing characters , i 'll take the latter
every time
-20.6927: carrying off a spot on scottish burr , duvall ( also a producer
) peels layers from this character that may well not have existed on paper
```

a. False positives: Some use positive words like “well loved” which may skew its weight. Some others make the sentence seem positive except for the twist at the end.
b. False negatives: Their comments don't hold such harsh negative words but they also don't have many positive words. They seem more neutral, but also long with many words. Which can be perceived as negative if most words are weighted negative

5. Again, using your new best model (in terms of dev error rate), predict the semi-blind test data, and submit it to Kaggle. What is your new error rate and ranking on the public leaderboard? Take a screenshot.



test.predicted.csv
Complete · 17s ago · fixed smart average
0.272

a. Average: Best dev err 26.3%, |w|=16744, time: 2.1 secs

3. Pruning the Vocabulary

1. Try neglecting one-count words in the training set during training. Did it improve the dev error rate?

a. Best dev err 25.9%, |w|=8425, time: 2.2 secs
b. It did improve the dev error rate slightly

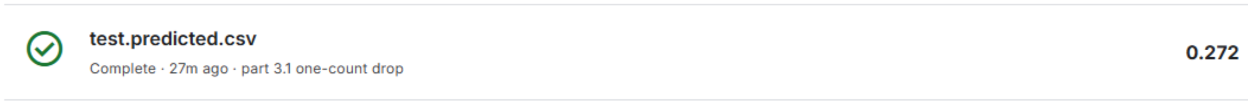
2. Did your model size shrink, and by how much?
    a. Dropped 7381 one-count words

3. Did update % change? Does the change make sense?
    a. 1.1% which means about 1% of training predictions were mistakes. I feel like this should be higher with how bad the error rate on tests is. Or maybe this means its way too over-fitted.

4. Did the training speed change?
    a. The speed did not really change

5. What about further pruning two-count words (words that appear twice in the training set)? Did it further improve dev error rate?
    a. Best dev err 26.6%, |w|=5934, update% = 1.3, time: 2.1 secs
    b. It seems to have made it worse. Maybe two-count words hold value in the training.

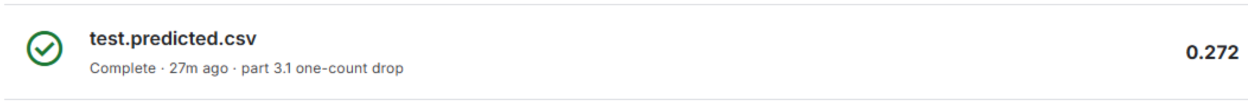
6. Using your current best model (in terms of dev error rate) from this part, predict the semi-blind test data, and submit it to Kaggle. What is your new error rate and ranking on the public leaderboard? Take a screenshot.


test.predicted.csv
Complete · 27m ago · part 3.1 one-count drop
0.272

4. Try some other learning algorithms with sklearn
    1. Which algorithm did you try? What adaptations did you make to your code to make it work with that algorithm? What specific setting (e.g., vocabulary pruning) did you use?
        a. I tried implementing KNN. I added the basic find best k function and altered the train and make_vector to handle x and y. I kept one-count pruning. It was significantly worse therefore i switch to experimenting with SVM.

    2. What's the dev error rate(s) and running time?
        a. Best dev err 26.6%, |model|=4359, update% = 100.0, time: 141.7 secs

    3. Submit your best prediction to Kaggle. What was your public score and rank? For example, our TA Zetian got ~ 24% (quite a bit better than 27%).

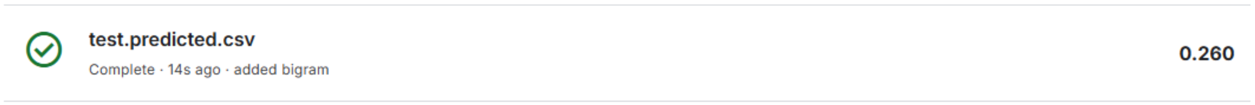

test.predicted.csv
Complete · 27m ago · part 3.1 one-count drop
0.272

        a. The KNN model was significantly worse therefore I switched to SVM and it was still slightly worse than the last perceptron version.

4. What did you learn in terms of the comparison between averaged perceptron and these other (presumably more popular and well-known) learning algorithms?
    a. The average perceptron has a strong advantage to this specific type of problem. To find positive or negative reviews in sentences with context.

5. Deployment
    1. What's the dev error rate(s) and how did you achieve it?
        a. Best dev err 25.1% at 61 epochs, |model|=25484, time: 231.1 secs
        b. Test error 49.8%
        c. The test error always seemed to get worse with more implementations but the dev error and the kaggle grade improved so that's what matters. I dont believe there is an issue with my predict function as it became a wrapper for the test.
        d. Used perceptron, average, shuffle, bigram, epoch step, one-count drop.

    2. Submit your best prediction to Kaggle. What was your overall best public score and final rank?


test.predicted.csv
Complete · 14s ago · added bigram
0.260

        a. 0.26, rank 26

6. Debriefing
    1. Approximately how many hours did you spend on this assignment?
        a. 16+ hr
    2. Would you rate it as easy, moderate, or difficult?
        a. Moderate - hard
    3. Did you work on it mostly alone, or mostly with other people?
        a. Alone
    4. How deeply do you feel you understand the material it covers (0%–100%)?
        a. I understand perceptrons a bit more but the alternate algorithms are sketchy like SVM.
    5. Any other comments?
        a. Finding notes for certain parts like one-count was a bit more difficult to find or understand how they should look at a coding level.