

## HW2

### Part 0: Sentiment Classification Task and Dataset

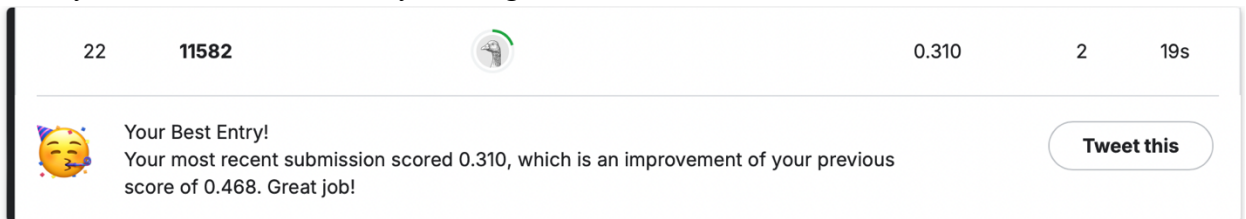
**Question: why is each of these steps necessary or helpful for machine learning?**

1. All words are lowercased
  - This would allow the model to treat the words like “great” and “Great” the same to avoid confusion and reduce the number of unique words
2. Punctuations are split from adjoining words and become separate “words”
  - Punctuations can carry meaning. This keeps punctuation separate so the model can understand their role
3. Verb contractions and the genitive of nouns are split into their component morphemes, and each morpheme is tagged separately
  - This breaks down words so the model can understand their parts better, making it easier to recognize patterns.
4. Single quotes are changed to forward- and backward- quotes (e.g., 'solaris' --> ` solaris ') and double quotes are changed to double single forward- and backward- quotes.
  - This makes quotes consistent and distinguishable to avoid confusion and help the model handle text better
5. There were a small amount of reviews in Spanish; I’ve removed them.
  - Keeps the dataset in one language (English) to make learning easier for the model.

### Part 1: Naïve Perceptron Baseline

1. **Take a look at `svector.py`, and briefly explain why it can support addition, subtraction, scalar product, dot product, and negation (0.5 pts).**
  - `Svector.py` is a sparse vector designed based on Python’s built-in default dictionary. The class contains functions to efficiently handle the operations. For example, addition and subtraction iterate over the dictionary’s key/value pairs and update the values accordingly.
2. **Take a look at `train.py`, and briefly explain `train()` and `test()` functions (0.5 pts).**
  - The `train()` function takes 3 inputs such as training file, development file, and epoch (5 by default). For each epoch, it goes through each example in the training set, creates an `svector` for the example, and updates the model if it makes an incorrect prediction. After each epoch, it test the model on development set while keeping track of the best dev error
  - The `test()` function takes the dev file and the testing model as inputs, it computes the error rate by counting the misclassifications and return the proportion (misclassification/total)
3. **There is one thing missing in my `train.py`: the bias dimension! Try add it. How did you do it? (Hint: by adding bias or `<bias>`?) Did it improve error on dev? (0.5 pts)**

- By adding  $v['\text{<bias>}'] = 1$  before returning the vector in the function `make_vector()`.  
The dev error is improved as the best dev error now is 28.9%
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. Hint: your public error rate should be ~ 31%. (0.5 pts)**
- My error rate is 31% and my ranking is 22.



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?? (0.5 pts)**
- Even though the dataset is balanced, the bias term is still important because it allows the model to adjust the decision boundary independently of individual features. This helps the model handle cases where the features alone aren't enough to make a clear distinction. The bias makes the model more robust by accounting for subtle variations or inconsistencies in the data that may not be perfectly separable by the features.

## Part 2: Average Perceptrons

1. **Train for 10 epochs and report the results. Did averaging improve the dev error rate? (Hint: should be around 26%). Did it also make dev error rates more stable? (1.5 pts)**

```
Epoch 1, updates 39.0%, dev error 31.4%
Epoch 2, updates 25.5%, dev error 27.7%
Epoch 3, updates 20.8%, dev error 27.2%
Epoch 4, updates 17.2%, dev error 27.6%
Epoch 5, updates 14.1%, dev error 27.2%
Epoch 6, updates 12.2%, dev error 26.7%
Epoch 7, updates 10.5%, dev error 26.3%
Epoch 8, updates 9.7%, dev error 26.4%
Epoch 9, updates 7.8%, dev error 26.3%
Epoch 10, updates 6.9%, dev error 26.3%
Best dev error 26.3%, |w|=15806, time: 70.7 secs
```

It did improve the error rate. My best error rate is 26.3%. The error rates don't seem to be stable here.

2. **Did smart averaging slow down training? (1 pt)**  
Yes I noticed it is quite slower for the average training (70.7s)
3. **What are the top 20 most positive and top 20 most negative features? Do they make sense? (1 pt)**  
**Top 20 Most Positive Features:**

engrossing: 975282.0000  
triumph: 906538.0000  
unexpected: 890237.0000  
rare: 889956.0000  
provides: 878381.0000  
french: 864314.0000  
skin: 849652.0000  
treat: 847015.0000  
pulls: 832015.0000  
culture: 819333.0000  
cinema: 819089.0000  
dots: 816579.0000  
wonderful: 813079.0000  
refreshingly: 804340.0000  
open: 791666.0000  
powerful: 780247.0000  
delightful: 778127.0000  
imax: 768587.0000  
smarter: 750566.0000  
flaws: 750394.0000

**Top 20 Most Negative Features:**

suffers: -765722.0000  
worst: -766857.0000  
scattered: -767592.0000  
problem: -770499.0000  
seagal: -772992.0000  
flat: -782628.0000  
neither: -787754.0000  
incoherent: -788491.0000  
unless: -794483.0000  
attempts: -795684.0000  
tv: -816025.0000  
instead: -817796.0000  
too: -846307.0000  
ill: -880309.0000  
fails: -891532.0000  
routine: -937094.0000  
badly: -949415.0000  
dull: -1030705.0000

generic: -1044776.0000

boring: -1193063.0000

This kind of makes sense since the top positive features correspond to words that show up in positive reviews and the same thing goes for negative features.

- 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? (2 pts)**

Top 5 Negative Examples Predicted as Positive:

Example: ' in this poor remake of such a well loved classic , parker exposes the limitations of his skill and the basic flaws in his vision ' | Prediction: 2808479.0000

Example: 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 | Prediction: 2405584.0000

Example: 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 | Prediction: 2377569.0000

Example: 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 | Prediction: 2296951.0000

Example: an atonal estrogen opera that demonizes feminism while gifting the most sympathetic male of the piece with a nice vomit bath at his wedding | Prediction: 1897881.0000

Top 5 Positive Examples Predicted as Negative:

Example: 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 | Prediction: -3488472.0000

Example: 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 | Prediction: -3245500.0000

Example: 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 | Prediction: -2323392.0000

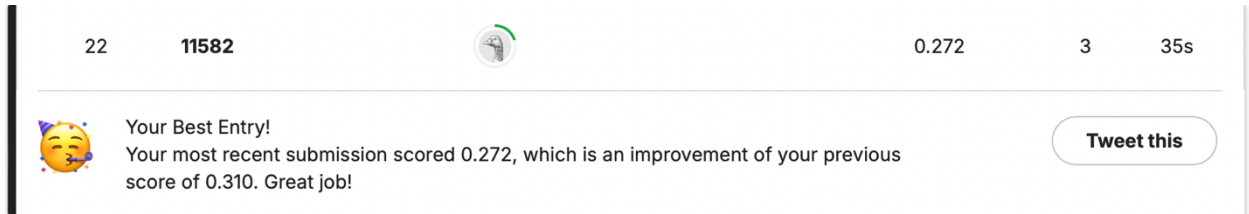
Example: if i have to choose between gorgeous animation and a lame story ( like , say , treasure planet ) or so so animation and an exciting , clever story with a batch of appealing characters , i 'll take the latter every time | Prediction: -2177851.0000

Example: carrying off a spot on scottish burr , duvall ( also a producer ) peels layers from this character that may well not have existed on paper | Prediction: -1655419.0000

**Observations:** the model seems to heavily rely on keywords (positive/negative sounding words) to determine whether a sentence is positive or negative. It gets confused with complex sentences, especially the ones with mixed keywords.

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 are your new error rate and ranking on the public leaderboard? Take a screenshot. Hint: your public error rate should improve to ~ 27%. (0.5 pts)

My new error rate on public leaderboard is 27.2% and my ranking is 22.



### Part 3: Pruning the Vocabulary

1. Try neglecting one-count words in the training set during training. Did it improve the dev error rate? (Hint: should help a little bit, error rate lower than 26%). (1 pt)

```
Epoch 1, updates 39.0%, dev error 31.6%
Epoch 2, updates 26.4%, dev error 27.5%
Epoch 3, updates 22.8%, dev error 26.8%
Epoch 4, updates 18.8%, dev error 26.6%
Epoch 5, updates 17.2%, dev error 25.9%
Epoch 6, updates 14.8%, dev error 26.5%
Epoch 7, updates 13.2%, dev error 27.0%
Epoch 8, updates 12.7%, dev error 26.7%
Epoch 9, updates 11.4%, dev error 26.6%
Epoch 10, updates 10.6%, dev error 26.2%
Best dev error 25.9%, |w|=8425, time: 38.1 secs
```

It did improve the dev error rate a little bit. My best error rate now is 25.9%.

2. **Did your model size shrink, and by how much? (Hint: should almost halve). Does this shrinking help prevent overfitting? (0.25 pts)**

my model size shrank to 8425. I think this shrinking helps prevent overfitting since it focuses more on frequent patterns and ignore the noises like words that appear only once.

3. **Did update % change? Does the change make sense? (0.25 pts) Did the training speed change? (0.25 pts)**

The update percentage was slightly higher in the pruned version, which makes sense because the model had fewer features to rely on after pruning leading to more mistakes early in training. However, the model still converged effectively. Training was significantly faster with pruning due to the reduced model size, cutting the time nearly in half from **70.7 seconds** to **38.1 seconds**. This improvement in speed is expected because fewer features (words) need to be processed.

4. **What about further pruning two-count words (words that appear twice in the training set)? Did it further improve dev error rate? (0.75 pts)**

```
Epoch 1, updates 38.9%, dev error 31.1%
Epoch 2, updates 28.2%, dev error 29.2%
Epoch 3, updates 23.7%, dev error 28.7%
Epoch 4, updates 21.7%, dev error 28.0%
Epoch 5, updates 18.5%, dev error 27.9%
Epoch 6, updates 17.7%, dev error 26.6%
Epoch 7, updates 16.2%, dev error 26.8%
Epoch 8, updates 15.2%, dev error 26.6%
Epoch 9, updates 14.1%, dev error 26.6%
Epoch 10, updates 12.6%, dev error 26.6%
Best dev error 26.6%, |w|=5934, time: 26.0 secs
```

This did not further improve the error rate but worsened it instead. This might be due to over pruning leading to the remove of important features.

5. **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 are your new error rate and ranking on the public leaderboard? Take a screenshot. Hint: your public error rate should still be ~ 27% and it is not necessarily lower than Part 2. (0.5 pts)**

My error rate is still about the same (27.2%) and my ranking is 23.

The screenshot shows a Kaggle leaderboard entry. At the top, the ranking is 23, the score is 0.272, and the time taken is 4m. Below this, a message says "Your Best Entry! Your most recent submission scored 0.272, which is the same as your previous score. Keep trying!"

## Part 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? (0.5 pts)


I used the Support Vector Classifier (SVC) algorithm from the sklearn.svm module with a linear kernel (kernel='linear') and a regularization parameter C=1. For vocabulary tuning, I ignored the words that appeared only once in the data. I also have negation handling by prefixing negation words with not.

2. What's the dev error rate(s) and running time? (0.5 pts)

```
SVC - Cross-validated accuracy: 76.96%
SVC - Dev Error Rate: 26.40%
SVC - Running Time: 30.38 seconds
SVC - Cross-validated Accuracy: 76.96%
```

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%). (0.5 pts)

11	11582		0.242	14	29s
----	-------	---	-------	----	-----



**Your Best Entry!**  
Your most recent submission scored 0.242, which is an improvement of your previous score of 0.264. Great job!

[Tweet this](#)

4. What did you learn in terms of the comparison between averaged perceptron and these other (presumably more popular and well-known) learning algorithms? (0.5 pts)

The averaged perceptron is fast and simple, updating with each example, but it's sensitive to data order and needs careful tuning. Algorithms like SVC and Logistic Regression usually perform better because they look at all data at once, finding boundaries that work well for the whole dataset


## Part 5: Deployment


1. What's the dev error rate(s) and how did you achieve it?

I tried **Logistic Regression** with **TF-IDF**. The dev error for this is 26.6%. I applied vocabulary pruning, which removed low-frequency words, reducing noise in the feature space. I also added a negation handling function to prefix words following negations helping the model interpret

sentiment accurately in the presence of negations. For the classifier, I chose Logistic Regression as it handles sparse, high-dimensional TF-IDF data effectively, leading to strong generalization on the development set.

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

9	11582		0.232	20	2m
---	-------	---	-------	----	----



Your Best Entry!  
Your most recent submission scored 0.232, which is the same as your previous score. Keep trying!

My best public score is 0.232 and rank 9.

## Part 6: Debriefing

**1. Approximately how many hours did you spend on this assignment?**

I spent about 7 hours on this homework

**2. Would you rate it as easy, moderate, or difficult?**

I would rate it as somehow difficult

**3. Did you work on it mostly alone, or mostly with other people?**

I did it mostly alone

**4. How deeply do you feel you understand the material it covers (0%–100%)?**

I feel I understand about 90%

**5. Any other comments?**

No comments