Sentiment Classification Task and Dataset

Q. Why is each of these steps necessary or helpful for machine learning?

Answer:

Text preprocessing involves essential steps to enhance the accuracy and effectiveness of natural language processing models. Firstly, breaking down verb contractions and genitive forms into their component morphemes ensures accurate tokenization and semantic capture. Secondly, lowercasing all words standardizes the text and reduces feature space dimensionality. Thirdly, treating punctuation marks as separate "words" aids sentiment and tone analysis. Consistent quotes formatting helps the model identify quotations. Finally, excluding non-English content, such as Spanish, is crucial when the model targets English text to maintain accuracy and relevance while preventing noise. These preprocessing steps collectively optimize the model's performance in understanding and analysing text data.

Naive Perceptron Baseline

Q. Take a look at svector.py, and briefly explain why it can support addition, subtraction, scalar product, dot product, and negation.

Answer:

Addition (__iadd__ and __add__ methods):

The __add__ Makes an new svector and it stores items from both of the vector by doing this it returns the new vector. It then represents the result of the addition.

The __iadd__ method (a += b) Adds up the values to the corresponding keys while iterating over the items in the (other) vector.

Subtraction (__sub__ and __isub__ methods):

The __sub__ method Uses the addition operation with a negated ‘other’ vector to perform its operations.

The __isub__ Updates the current vector by using addition operation along with a the negated ‘other’ vector.

Scalar Product (__mul__ and __rmul__ methods):

The __mul__ Using every element of the vector by a scalar ‘c’ it multiplies the elements. It will then return a new vector that has the scaled values.

The __rmul__ Is used to manage the scalar-vector multiplication in the reverse order

Dot Product (dot method):

The dot Calculates the dot product between two vectors.The dot product iterates over the vectors multiplying each element by their respective large vector and returns the sum.

Negation (__neg__ method):

The __neg__ Makes a new svector and it gives negation to each of the elements that belong to the original vector.

Q. Take a look at train.py, and briefly explain train() and test() functions.

Answer:

train(trainfile, devfile, testfile, epochs=5):

This function uses a training dataset to train a linear binary classification model. Next it evaluates the performance of the model using a development dataset. The number of training iterations is controlled by the epochs. It is used to initialize a model vector as an empty svector. The best development errors encountered during the training are kept track of. At the end after every epochs is executed, it reports the best development error the training time, the size of the averaged model, and the best development error.

test(devfile, model):

This function uses a development model and evaluates the performance of a model. Using the given model, it reads data from the development file and gives the error rate. It returns the error rate in a decimal for ranging between 0 to 1 and lower the value the better it is.

Q: There is one thing missing in my train.py: the bias dimension! Try to add it. How did you do it? (Hint: by adding bias or ?) Did it improve error on dev?

Answer:

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

It has improved the error on dev.

Q: 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??

Answer:

The bias dimension improves the flexibility and it helps to get more complex relationships between the words and sentiment labels. The bias dimension needs to be present because sentiment analysis has more to it than just the presence and absence of any particular words. The bias word denotes a baseline or offset prediction that is independent of the input information. This is especially significant when our model must account for instances in which none of the observed words have a large influence on sentiment but there is still an overall sentiment tendency. When we want to use our model for sentiment analysis on new, previously unknown text, the bias term helps the algorithm to make decent predictions even for language that did not appear in the training data. Without a bias term, your model may struggle with text that lacks feature values. In reality, introducing a bias word can improve your model's overall performance and prediction skills, making it more robust and capable of dealing with a variety of linguistic situations.

Average Perceptron

Q. 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?

Answer:

Yes, it did improve the error rate and it did make the dev error rates more stable.

epoch 1, update 39.3%, dev 29.6%

epoch 2, update 25.2%, dev 27.5%

epoch 3, update 20.6%, dev 27.7%

epoch 4, update 17.1%, dev 26.4%

epoch 5, update 14.4%, dev 26.2%

epoch 6, update 12.5%, dev 26.2%

epoch 7, update 10.3%, dev 26.2%

epoch 8, update 9.8%, dev 26.5%

epoch 9, update 8.4%, dev 26.8%

epoch 10, update 7.7%, dev 26.6%

best dev err 26.2%, |w|=15805, time: 215.9 secs

Q. Did smart averaging slow down training?

Answer:

Yes.

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

Answer:

Top 20 most positive features:

- rare
- treat
- open
- am
- smarter
- 1920
- cinema
- imax
- remarkable
- powerful
- unexpected
- delightful
- engrossing
- provides
- martha
- refreshingly
- manages
- speaks
- pulls
- triumph

Top 20 most negative features:

- god
- ill
- animal
- fails
- instead
- tv
- unless
- lacking
- worse
- pie
- inane
- routine

- generic
- scattered
- incoherent
- flat
- boring
- problem
- dull
- badly

Yes, it makes sense.

Q. 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?

Answer:

5 negative examples in dev where my model most strongly believes to be positive:

+ the banter between calvin and his fellow barbers feels like a streetwise mclaughlin group and never fails to entertain

+ god bless crudup and his aversion to taking the easy hollywood road and cashing in on his movie star gorgeousness

+ devotees of star trek ii the wrath of khan will feel a nagging sense of deja vu , and the grandeur of the best next generation episodes is lacking

+ good actors have a radar for juicy roles there 's a plethora of characters in this picture , and not one of them is flat

+ like these russo guys lookin ' for their mamet instead found their sturges

5 positive examples in dev where my model most strongly believes to be negative:

- well meaning to a fault , antwone fisher manages the dubious feat of turning one man 's triumph of will into everyman 's romance comedy

- if the last man were the last movie left on earth , there would be a toss up between presiding over the end of cinema as we know it and another night of delightful hand shadows

- spirit is a visual treat , and it takes chances that are bold by studio standards , but it lacks a strong narrative

- the laughs are as rare as snake foo yung

- yes , spirited away is a triumph of imagination , but it 's also a failure of storytelling

Observations:

- I observed that the sentiment analysis lacks on detecting sentences with irony or sarcasm.

- It is highly context driven.

Pruning the Vocabulary

Q. 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%).

Answer:

Yes, but very little.

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

Yes, it did shrink. The new model size is 8425. We could reduce the capacity of the model it will the help prevent overfitting.

Q. Did update % change? Does the change make sense?

The change does make sense as now it makes more updates on the next epochs than before indicating training on denser datasets.

Q. Did the training speed change?

Answer:

It has decreased.

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

Answer:

Dev error rate was very similar to pruning one-count words. The training speed decreased.

Try some other learning algorithms with sklearn

Q. Which algorithm did you try? What adaptations did you make to your code to make it work with that algorithm?

Answer:

The algorithm I tried was SVM. The adaptations I took were:

- Converting svector objects to numpy vectors
- Using SVC with a linear kernel
- Data preprocessing
- Pruning out one-count & two-count words

Q. What's the dev error rate(s) and running time?

Answer:

Development error: 28.60%

Time taken: 123.10 seconds

Development error: 29.30%

Time taken: 90.06 seconds

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

Answer:

Comparing both of them the training time for average perceptron is faster than the SVM. The error rates on dev for both of them are great.

Deployment

Q. What's your best error rate on dev, and which algorithm and setting achieved it?

Answer:

Best error rate on dev - 26.0 %

Algorithm - Perceptron

Setting - Neglect one-count words, using the whole train dataset.

Debrief

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

    I gave 1 hours every day to the course and on the assignment.

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

    Moderate

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

    Mostly with other people

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

    75%