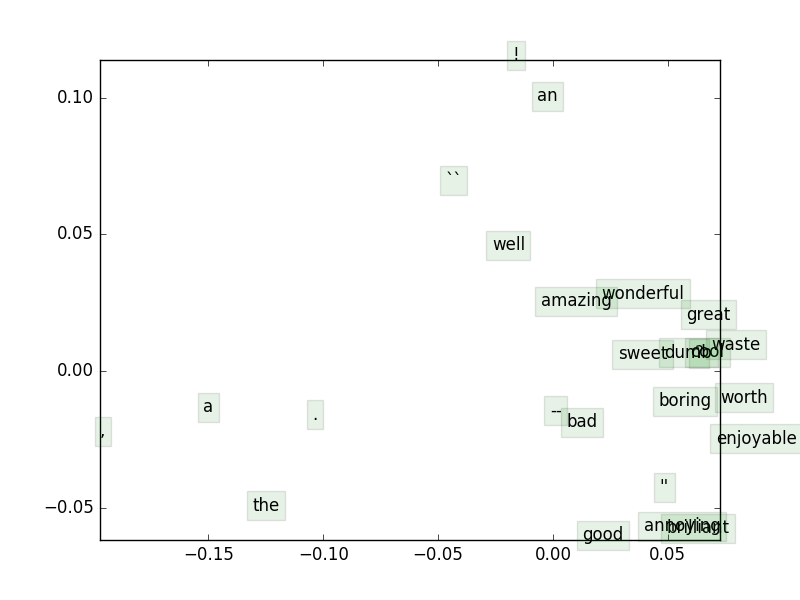


Prob 3.g



Most of the words are on the bottom right corner. Some very useful words and notations such as “a”, “the” is rather far away to the majority.

Prob 4.b

Use of regularization is to avoid overfitting.

Prob 4.c

def chooseBestModel(results):

"""Choose the best model based on parameter tuning on the dev set

Arguments:

results -- A list of python dictionaries of the following format:

{

"reg": regularization,

"clf": classifier,

"train": trainAccuracy,

"dev": devAccuracy,

"test": testAccuracy

}

Returns:

Your chosen result dictionary.

"""

bestResult = results[0]

### YOUR CODE HERE

for i in range(len(results)):

currDict = results[i]

if currDict["test"] > bestResult["test"]:

bestResult = currDict

### END YOUR CODE

return bestResult

Prob 4.d

The result based on my trainin:

Reg Train Dev Test

0.00E+00 31.016 32.516 30.407

0.00E+00 31.016 32.516 30.407

1.00E+00 28.897 29.609 27.149

1.00E+01 27.247 25.522 23.077

1.00E+02 27.247 25.522 23.032

1.00E+03 27.247 25.522 23.032

1.00E+04 27.247 25.522 23.032

1.00E+05 27.247 25.522 23.032

1.00E+06 27.247 25.522 23.032

1.00E+07 27.247 25.522 23.032

1.00E+08 27.247 25.522 23.032

1.00E+09 27.247 25.522 23.032

The result based on pre-trained:

0.00E+00 39.923 36.421 37.059

0.00E+00 39.923 36.421 37.059

1.00E+00 39.525 36.603 37.330

1.00E+01 38.624 36.876 37.692

1.00E+02 36.330 35.059 35.701

1.00E+03 32.163 31.153 30.588

1.00E+04 27.271 25.613 23.122

1.00E+05 27.247 25.522 23.032

1.00E+06 27.247 25.522 23.032

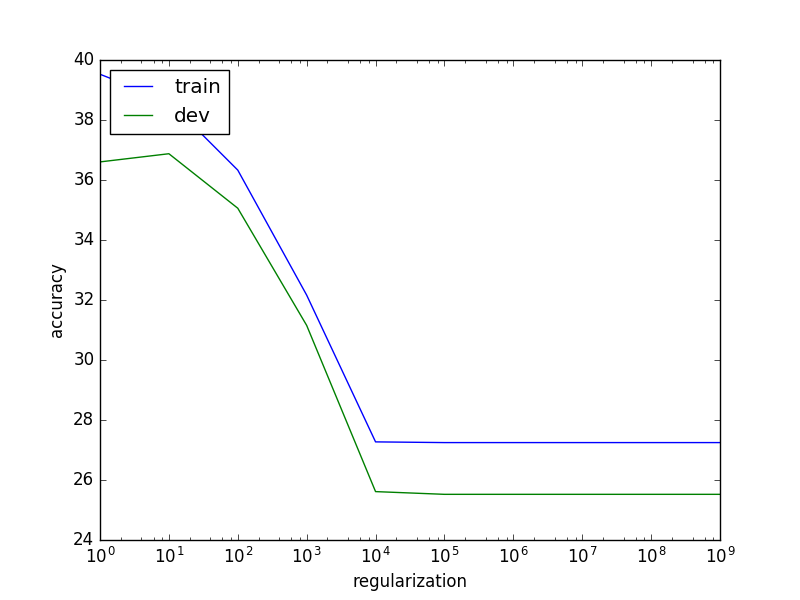
1.00E+07 27.247 25.522 23.032

1.00E+08 27.247 25.522 23.032

1.00E+09 27.247 25.522 23.032

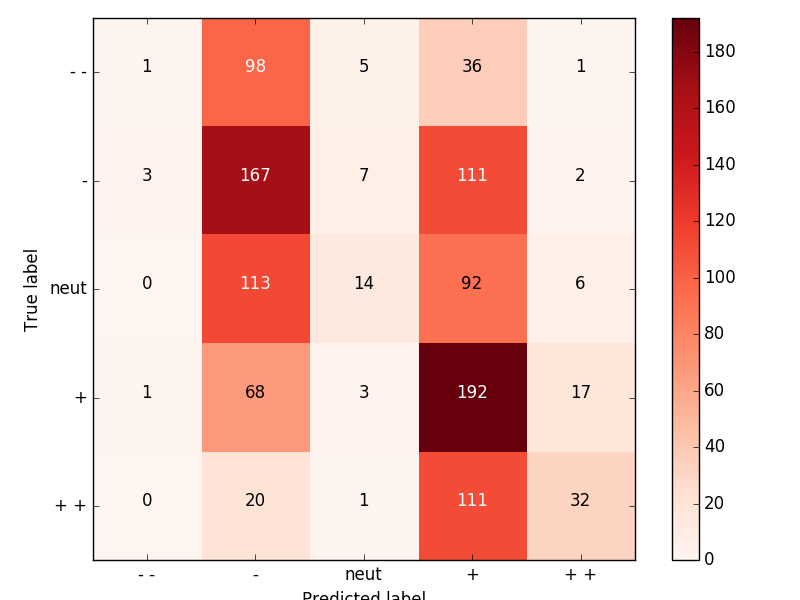
In most cases the training accuracy is better than dev accuracy, and the dev accuracy is better than test accuracy. The pre-trained result is better than mine because it is using GloVe vectors (which may be better than word2vec) with a bigger dataset (Wikipedia data). The pre-trained data may also trained for a longer period.

Prob 4.e



The training accuracy is always decreasing when the regularization factor is increasing until it reached a platform. However, the dev accuracy increased a little bit at first, but then also decreased.

Prob 4.f



From the diagram, we can see that the algorithm tends to giving “+” or “—”. Overall it gives a pretty acceptable result on these two labels, but not very good on “—— ”, “++”, and neutual.

Prob 4.g

Example 1.

*3 1 whether you like rap music or loathe it , you ca n't deny either the tragic loss of two young men in the prime of their talent or the power of this movie .*

Here may be the program mistakenly only gives a large weight on strong words like “deny”, “tragic”, but missed words like “talent”.

Example 2.

*4 1 the movie is n't just hilarious : it 's witty and inventive , too , and in hindsight , it is n't even all that dumb .*

Although, words like “dumb” is a strong signal of bad sentiment, but the algorithm may missed considering “isn’t”.

Example 3.

*0 1 it 's like watching a nightmare made flesh .*

The length of the sentence could also be taken into account.