```
#######################
# TESTING NAIVE BAYES #
#######################
c(k) values:
trump spoke 637 statements
clinton spoke 455 statements

c(k, w) values:
trump said the word country 161 times.
trump said the word president 39 times.
clinton said the word country 84 times.
clinton said the word president 182 times.

p(k) values:
P(trump): 0.20183776932826364
P(clinton): 0.14416983523447402

p(w | k) values:
P(country|trump): 0.004146301915637812
P(president|trump): 0.0010063339844906174
P(country|clinton): 0.0015826447291908564
P(president|clinton): 0.0034268680759293097

p(k | d) values:
walker: 0.05819253562129055
webb: 0.0584913342408921
bush: 0.057519211957027985
sanders: 0.05867792306489983
o'malley: 0.05910086665662609
kasich: 0.05851447242677072
rubio: 0.05745483478687285
clinton: 0.05773524776977021
huckabee: 0.05859409673229641
fiorina: 0.058878845947172735
chafee: 0.05997099305474091
cruz: 0.058178141540635275
```

```
christie: 0.058175485469083375
carson: 0.05860566028333849
paul: 0.057817009733424136
perry: 0.06558603197053227
trump: 0.05850730874462613
253 correct out of 400
```

Implementation Choices:
I did add n smoothing, with n = 0.1. This type of smoothing resulted in the largest accuracy on dev.
I also had to adjust the proportional probability values by a constant multiplicitive factor to avoid underflow when taking the exponential

```
#################################
# TESTING LOGRITHMIC REGRESSION #
#################################
Iteration number: 1
Negative log likelihood: 799.0586638276659
Accuracy on dev: 0.41

Iteration number: 2
Negative log likelihood: 785.2813466798242
Accuracy on dev: 0.4025

Iteration number: 3
Negative log likelihood: 682.2998011794491
Accuracy on dev: 0.4725

Iteration number: 4
Negative log likelihood: 705.2748682360038
Accuracy on dev: 0.44

Iteration number: 5
Negative log likelihood: 633.3150473923641
Accuracy on dev: 0.5025
```

```
Iteration number: 6
Negative log likelihood: 630.1377523682189
Accuracy on dev: 0.485

Iteration number: 7
Negative log likelihood: 626.9399058356508
Accuracy on dev: 0.4975

Iteration number: 8
Negative log likelihood: 616.3110702230416
Accuracy on dev: 0.4825

Iteration number: 9
Negative log likelihood: 600.8463095570114
Accuracy on dev: 0.5025

Iteration number: 10
Negative log likelihood: 616.1096852944954
Accuracy on dev: 0.52

Iteration number: 11
Negative log likelihood: 608.3354676838235
Accuracy on dev: 0.515

Iteration number: 12
Negative log likelihood: 621.5521970900686
Accuracy on dev: 0.5

Iteration number: 13
Negative log likelihood: 624.718968514814
Accuracy on dev: 0.51

Iteration number: 14
Negative log likelihood: 607.4172649813022
Accuracy on dev: 0.53
```

```
Iteration number: 15
Negative log likelihood: 609.7039090412981
Accuracy on dev: 0.51

Iteration number: 16
Negative log likelihood: 612.8971166694171
Accuracy on dev: 0.5075

Iteration number: 17
Negative log likelihood: 605.3068448759975
Accuracy on dev: 0.5175

Iteration number: 18
Negative log likelihood: 618.2739950425364
Accuracy on dev: 0.5225

Iteration number: 19
Negative log likelihood: 616.4247975982305
Accuracy on dev: 0.53

Iteration number: 20
Negative log likelihood: 606.9670215203645
Accuracy on dev: 0.5175

Iteration number: 21
Negative log likelihood: 616.6155024613658
Accuracy on dev: 0.5125

Iteration number: 22
Negative log likelihood: 615.1025311686248
Accuracy on dev: 0.5175

Iteration number: 23
Negative log likelihood: 609.7559437671937
Accuracy on dev: 0.515
```

```
Iteration number: 24
Negative log likelihood: 626.9843737935621
Accuracy on dev: 0.5075

Iteration number: 25
Negative log likelihood: 620.2846671700453
Accuracy on dev: 0.51

Iteration number: 26
Negative log likelihood: 619.8495196818235
Accuracy on dev: 0.515

Iteration number: 27
Negative log likelihood: 617.7727569566072
Accuracy on dev: 0.5125

Iteration number: 28
Negative log likelihood: 630.7252221852475
Accuracy on dev: 0.5025

Iteration number: 29
Negative log likelihood: 620.9905533502584
Accuracy on dev: 0.52

Iteration number: 30
Negative log likelihood: 619.7400946745261
Accuracy on dev: 0.51

λ(k) values:
λ(trump): 1.811058394753708
λ(clinton): 0.85780033801669

λ(k, w) values:
λ(trump, country): 0.5188703489575319
λ(trump, president): -0.3911166931818157
λ(clinton, country): -0.2400772971139224
```

λ(clinton, president): 0.5942518321346586

P(k | d) for the first line of dev:

```
{'walker': 0.007831911876007996, 'paul': 0.031159035342257 35, 'webb': 0.0011562974702490983,
'bush': 0.09957160276848613, "o'malley": 0.009050160771859526, 'sanders': 0.028223624005591603,
'kasich': 0.01016595212721458, 'rubio': 0.08069665723916244, 'clinton': 0.025011260199564872,
'huckabee': 0.007278893161164184, 'chafee': 0.003955074855651767, 'trump': 0.5712331659863326,
'christie': 0.010660465519309818, 'carson': 0.004937984107092531, 'fiorina': 0.03395221390153568,
'perry': 9.282039166782065e-05, 'cruz': 0.07502288027685193}
```

Accuracy on test: 0.565

Implementation Choices:

I randomly shuffled the training lines before each iteration, as well as each test set I tested on

I started with a learning rate of 0.1. This allowed for quick increases in accuracy initially, but I decreased this value by 5% each iteration. This made the steps smaller towards the end.

I chose 30 iterations because at this point, the learning rate is small enough that the model should be hovering around its maximum. Note: .95^30 ~= 20% of the original learning rate.

for λ(k), I assumed there was a dummy word (the empty string in my case) that occurred once per document.