mattholtzer9@berkeley.edu ¶

In [14]:

```
output:

url: http://boingboing.net/#85534171
 date: not supplied

 arts and letters daily, a wonderful and dense blog, has folded up its tent due
 to the bankruptcy of its parent company. a&l daily will be auctioned off by the
 receivers. link[1] discuss[2] (_thanks, misha!_)

 [1] http://www.aldaily.com/
 [2] http://www.quicktopic.com/boing/h/zlfterjnd6jf



<html>
 <head>
 </head>
 <body>
 <font size=3d"4"><b> a man endowed with a 7-8" hammer is simply<br>
  better equipped than a man with a 5-6"hammer. <br>
 <br>would you rather have<br>more than enough to get the job done or fall =
 short. it's totally up<br>to you. our methods are guaranteed to increase y=
 our size by 1-3"<br> <a href=3d"http://209.163.187.47/cgi-bin/index.php?10=
 004">come in here and see how</a>
 </body>
 </html>
```

The ham one includes a specific name and the spam one does not. When emails are from somebody that knows you it is probably more likely that they will address you in a way that shows that either with a gender marker or personal use of a name, which somebody sending mass spam is less likely to have.

1. The sensitivity is 0%. The specificity is 100%.
2. The accuracy is the percentage of ham emails in the training set, about 74.365%
3. A classifier whose accuracy is basically just as good as literally just picking the more likely option for every one, your classifier hasn't really accomplished anything. In this case especially it's probably worse. Since ultimately the point of marking spam is to filter spam somehow then I'd much rather everything go into my inbox than getting some spam out and losing some of my good ol' ham to the filter too.
4. The sensitivity is 16.45%. The specificity is 96.13%. We are much more likely to have false negatives when we see spam versus false positives when we see ham. This makes sense because our classifier is running off of very little information and is forced to apply ham to most things when it is dealt with a lack of information.
5. They are all medical related. This isn't a huge range of words. Most of the emails don't even contain any of these words and thus our classifier can't really do much with them whatsoever.

1. I decided to go with the method of adding more words to train the set on. I attempted a variety of things. Before really even starting anything I wanted to play around with where my intuition took me, use words that I know or think to be common in spam. One thing I did was look up lists of extremely common words and phrases in spam emails in addition to looking through parts of the test data set to see what kind of wording was in those. Given that info I tactically added different words that came across to me as highly associated with spam or not, ranging from different types of greetings to header text to actual content. This successfully got me up to around 93% or so pretty quickly. After that I moved onto some slightly more sophisticated analysis to try and find words to add in a more targeted way. You can see how that process went by reading through the descriptions in my EDA cells below.

2. One interesting thing I tried that did not work was running a test that looped through the entire list I had up to one point, around 200 or so, and removed any that lowered the score value. This took nearly an hour to run, only resulted in a couple words being removed, didn't increase the score by much, and had literally zero effect on the kaggle submission. I realized after that this was a bit of a silly attempt. Testing how removing things results in being able to fit the model on the training data doesn't matter that much past a certain point. You can get super focused on getting that perfect but really that just results in overfitting the model to the training data, which doesn't really help with predictions. I found much more success in focusing on the general idea of finding spam and ham in a more generalized sense during this project than to focus too closely on minor discrepencies in the training data.

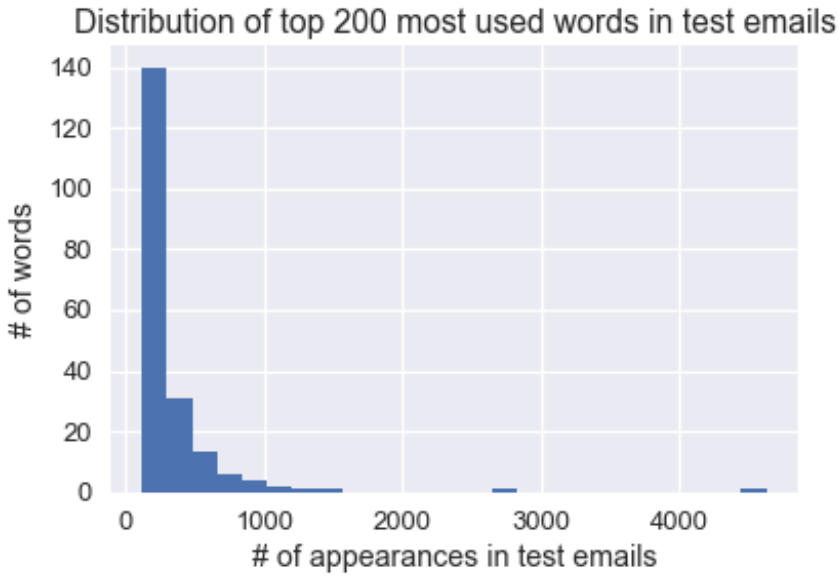
3. I'm surprised by how effective just doing a word based regression prediction can be. Getting up to that first 85% correct or so isn't really that hard and getting up to even 95%+ is fairly doable just by adding words as features. I'm sure that getting towards 99/100% requires some more complicated analysis but getting pretty good isn't that hard with some basic data science and regressions. My code got up to 98.2% on Kaggle without doing any more than training my model on a bunch of words. Cleverly chosen words, but words nonetheless.

In [312]:

```
output:
```

Out[312]:

```
<matplotlib.text.Text at 0x1733334e0>
```

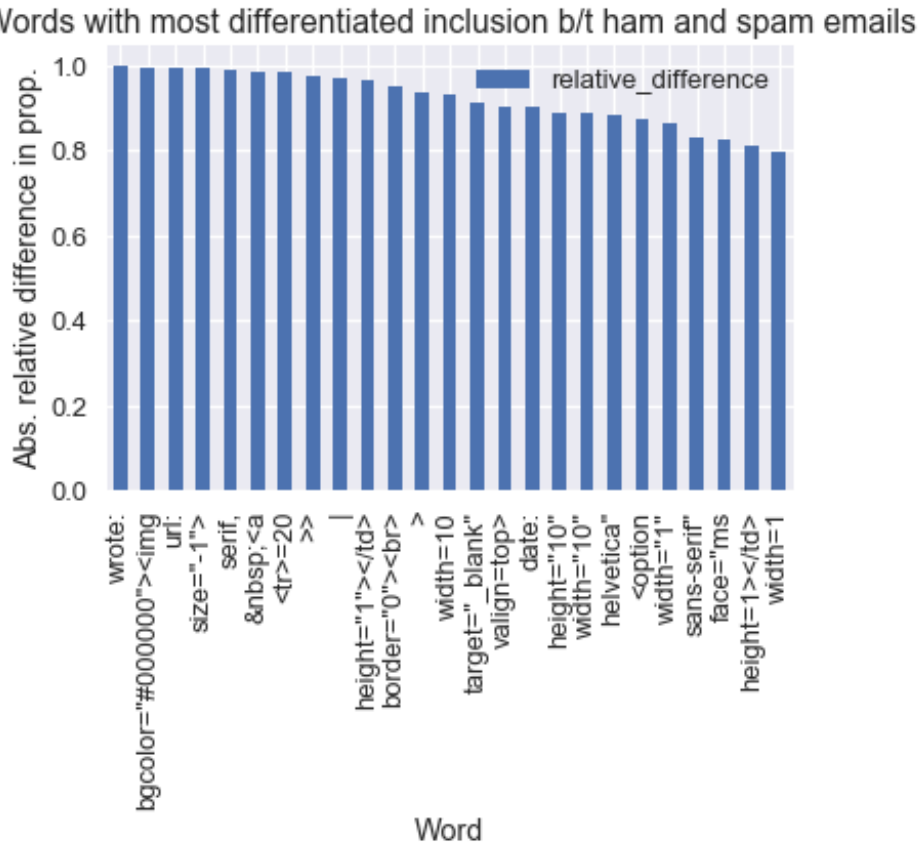


In [336]:

```
output:
```

Out[336]:

```
<matplotlib.text.Text at 0x172a74e48>
```

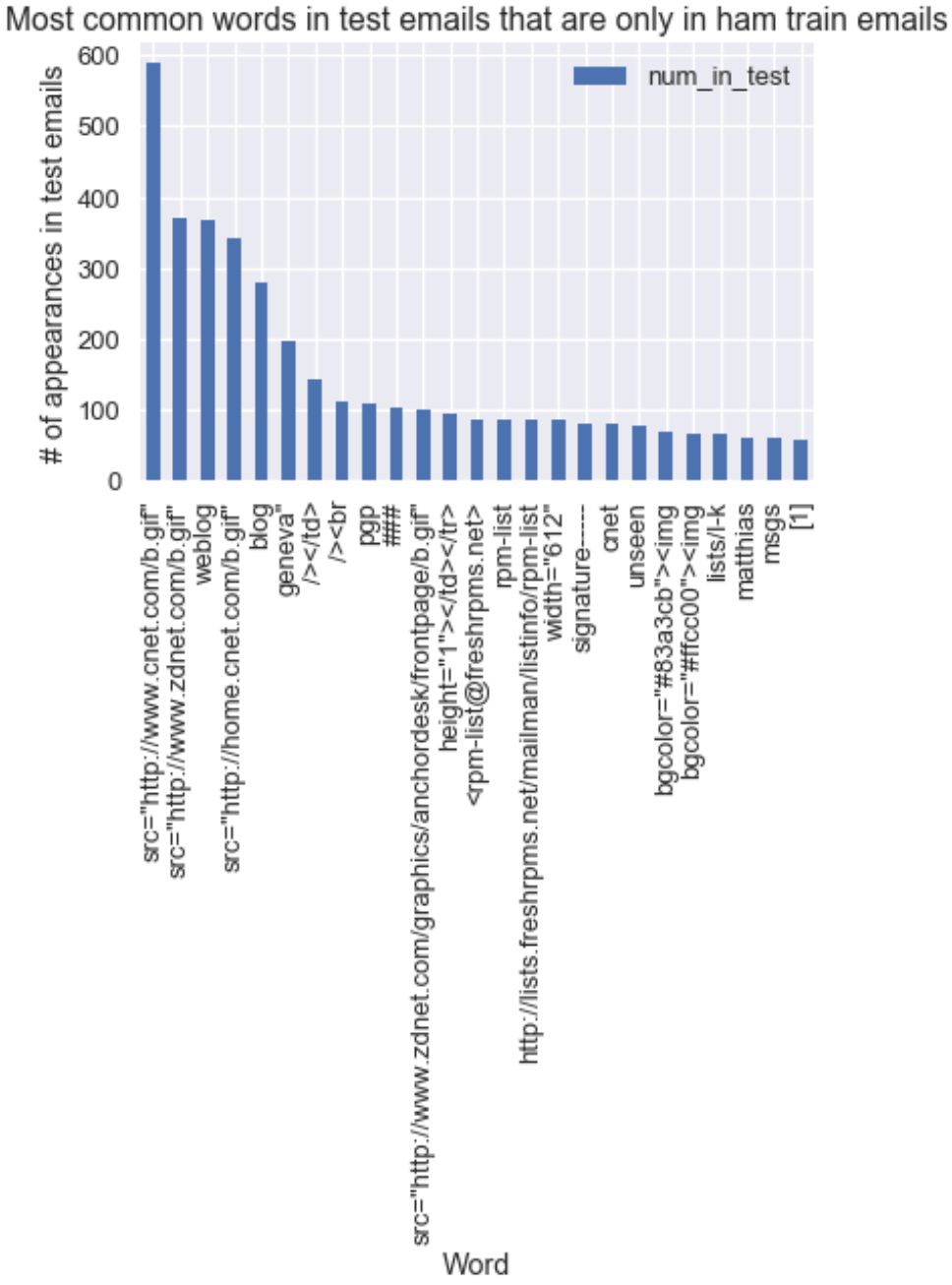


In [296]:

```
output:
```

Out[296]:

```
<matplotlib.text.Text at 0x193d44fd0>
```

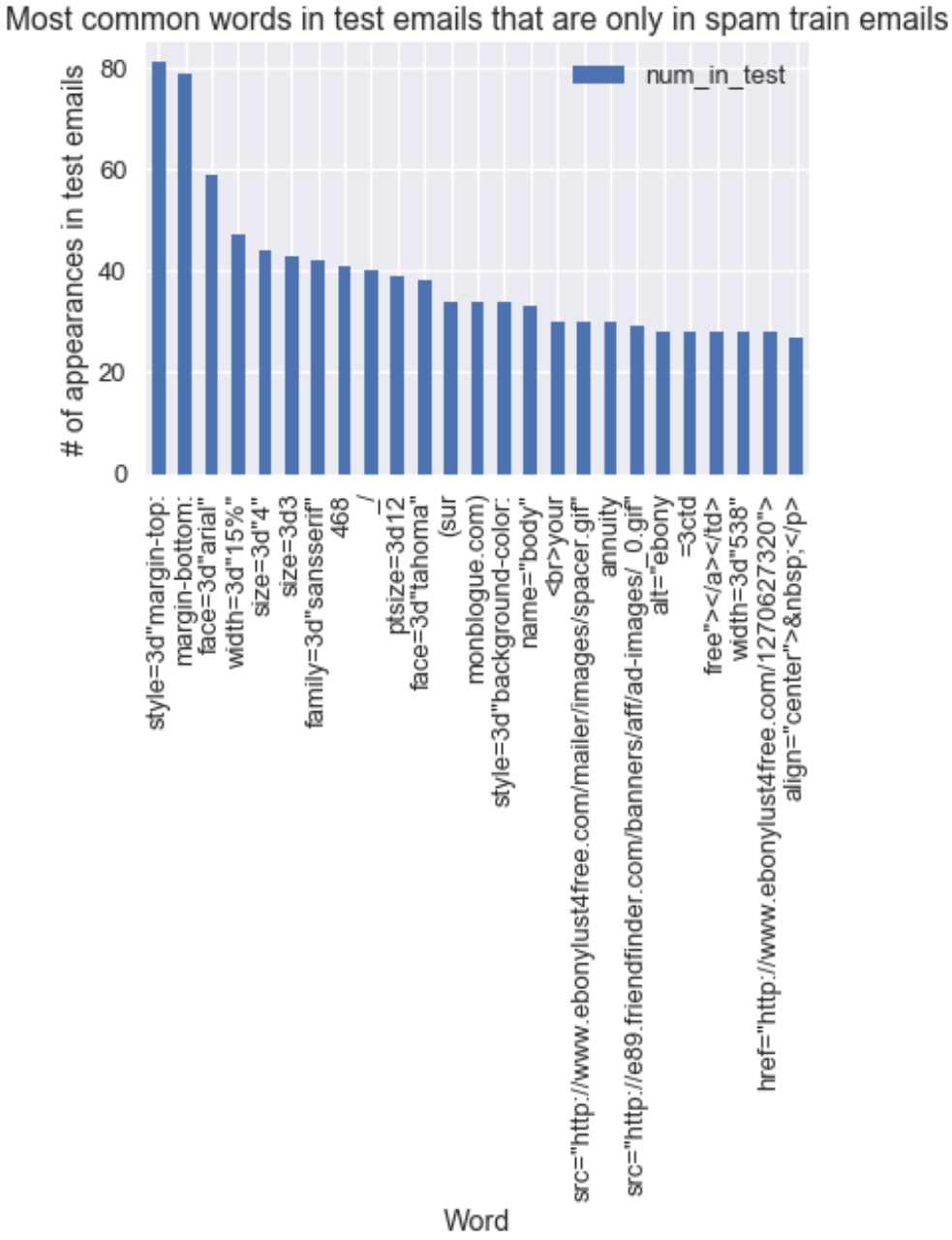


In [295]:

```
output:
```

Out[295]:

```
<matplotlib.text.Text at 0x196381470>
```



In [291]:

output: