Notebook

April 20, 2020

0.0.1 Question 1c

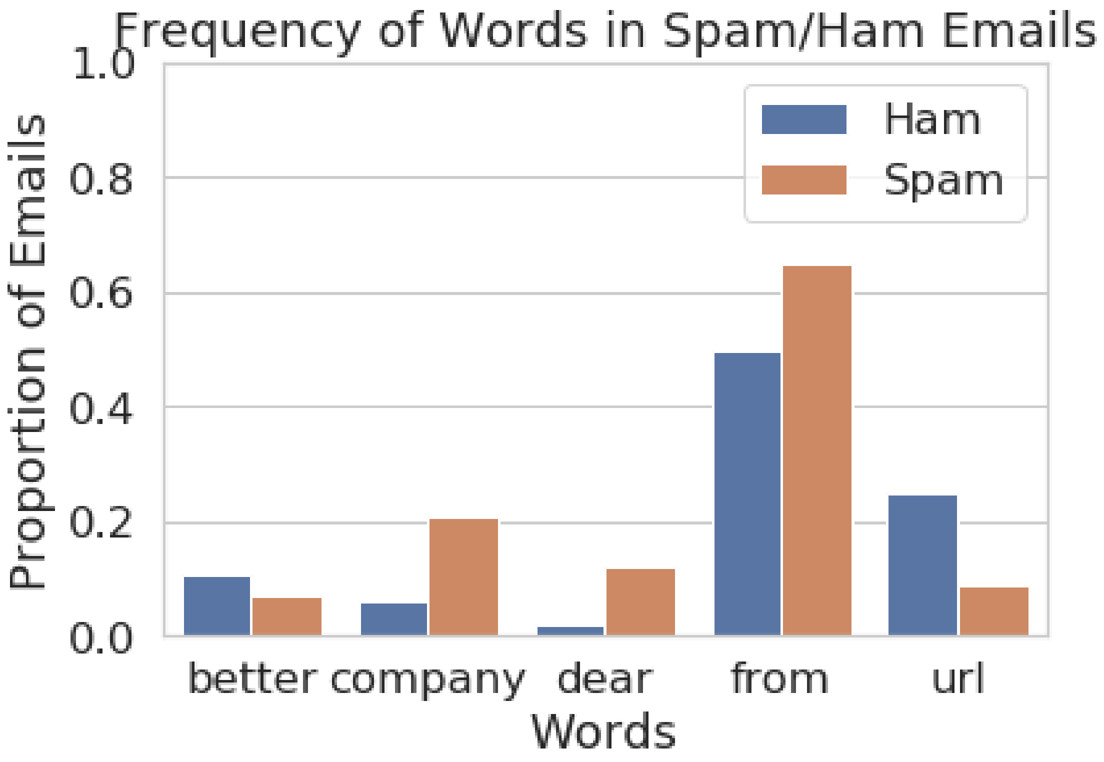
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

The difference between the two email is that the one that is ham consists of proper form with clear message that is readable by whoever recieves the email. However, the email for the email that is identified as spam is written in an html form which may be confusing for the person who doesn't know html if they recieve the email.

0.0.2 Question 3a

Create a bar chart like the one above comparing the proportion of spam and ham emails containing certain words. Choose a set of words that are different from the ones above, but also have different proportions for the two classes. Make sure to only consider emails from `train`.

```
In [13]: # train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of em

         # train_spam = train.query("spam == 1")
         # train_ham = train.query("spam == 0")

         # choice_of_words = ["url", "from", "this", "on", "dear", "n", "with"]
         # spam = pd.DataFrame(words_in_texts(choice_of_words, train_spam.iloc[:,2]))
         # for i in np.arange(0,len(choice_of_words)):
         #     spam = spam.rename(columns={i:choice_of_words[i]})
         # spam["type"] = "spam"

         # ham = pd.DataFrame(words_in_texts(choice_of_words, train_ham.iloc[:,2]))
         # for i in np.arange(0,len(choice_of_words)):
         #     ham = ham.rename(columns={i: choice_of_words[i]})
         # ham["type"] = ham
         # mix = pd.concat([ham, spam])
         # melted_down = mix.melt("type")
         # plt.figure(figsize=(10,10))
         # sns.barplot(x="variable", y="value", hue="type", data=melted_down, ci=none);

         # here = pd.DataFrame({"url" : list(train.email.str.contains("url")), "from":list(train.email.
         # here = here.replace(True, 1)
         # here['type'] = here['type'].replace(0, 'ham')
         # here['type'] = here['type'].replace(1, 'spam')
         # melted_down = here.melt('type')
         # melted_down = melted_down.rename(columns={'variable':'Words'})
         # # melted_down["Proportion of Emails"] = list(melted_down.groupby("Words").agg('sum') / len(m
         # grouping = melted_down.groupby("Words").agg('sum') / len(here)
         # # merge = melted_down.append(grouping)
         # # sns.barplot(x='Words', y='Proportion of Emails')
         # # grouping

         train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai

         chosen_words = ['better', 'from', 'url', 'company', 'dear']
         values = words_in_texts(chosen_words, train['email'])

         data_frame = pd.DataFrame(data = values, columns = chosen_words)
         data_frame['label'] = train['spam']

         ham_spam = data_frame.replace({'label' : {0: 'Ham', 1: 'Spam'}})
         melted = ham_spam.melt('label').groupby(['label', 'variable']).mean().reset_index()
         # melted

         sns.barplot(x = 'variable', y = 'value', hue = 'label', data = melted)

         plt.title('Frequency of Words in Spam/Ham Emails')
         plt.ylabel('Proportion of Emails')
         plt.xlabel('Words')
plt.legend(title = '')
plt.ylim([0,1]);
```



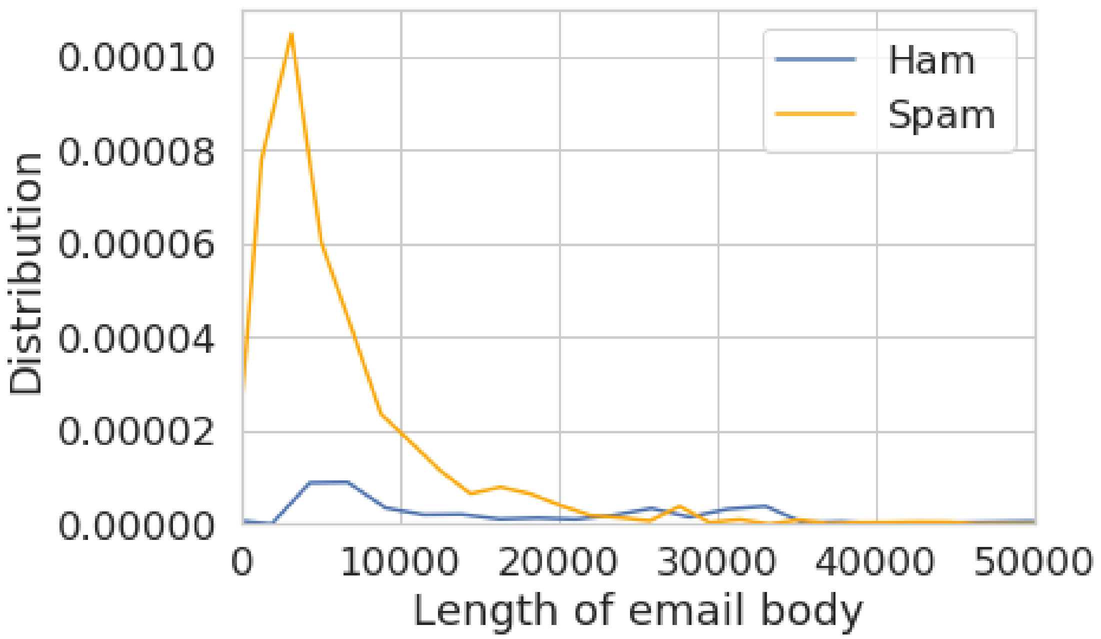
0.0.3 Question 3b

Create a *class conditional density plot* like the one above (using `sns.distplot`), comparing the distribution of the length of spam emails to the distribution of the length of ham emails in the training set. Set the x-axis limit from 0 to 50000.

```
In [14]: hams = train[train['spam'] == 0]
         spams = train[train['spam'] == 1]

         ham_train = []
         for ham in range(len(hams)):
             ham_trained = len(hams['email'].iloc[ham])
             ham_train.append(ham_trained)

         hams['length'] = ham_train
         sns.distplot(hams['length'], hist = False, color='b', label = 'Ham')
         plt.xlim(left = 0, right = 50000)

         spam_train = []
         for spam in range(len(spams)):
             spam_trained = len(spams['email'].iloc[spam])
             spam_train.append(spam_trained)

         spams['length'] = spam_train
         sns.distplot(spams['length'], hist = False, color = "orange", label = 'Spam')
         plt.xlabel('Length of email body')
         plt.ylabel('Distribution')
         plt.xlim(left = 0, right = 50000)
         plt.legend()
```

```
/srv/conda/envs/data100/lib/python3.7/site-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
… Omitting 5 lines …

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
```

```
Out[14]: <matplotlib.legend.Legend at 0x7f166928ca50>
```


0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
Distribution
0
10000
20000
30000
40000
50000
Length of email body
Ham
Spam