Notebook

November 24, 2019

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.

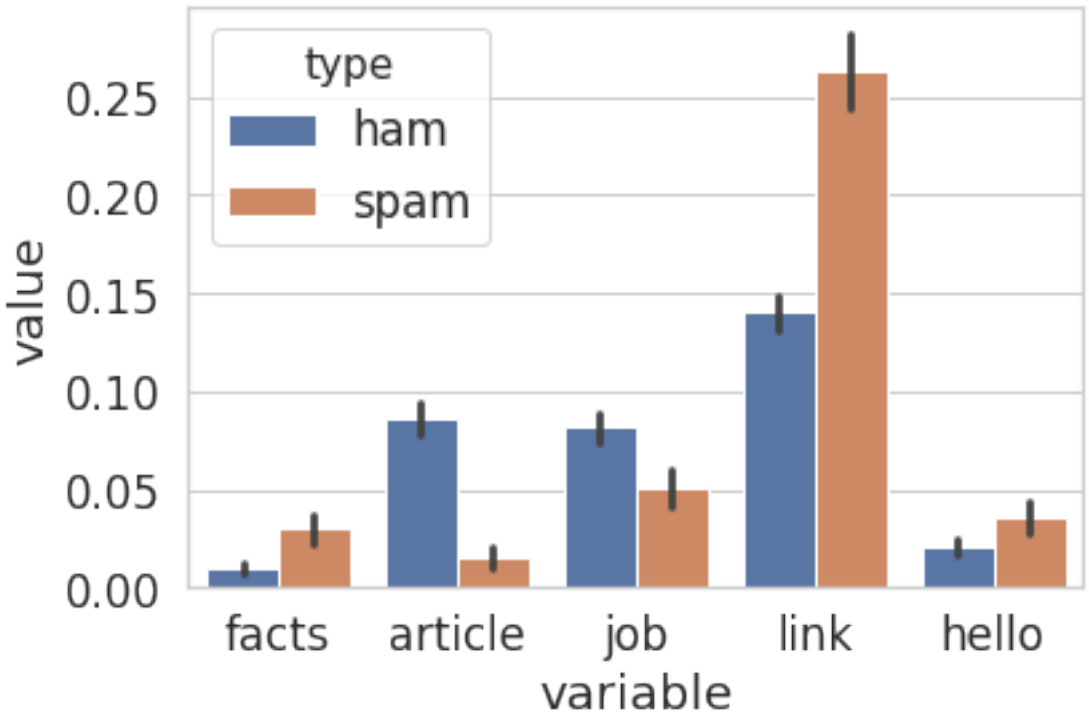
Plain text vs a html document for the different between ham vs spam and more links in ham

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 emai

words = ['facts','article','job','link','hello']
d = {0:'ham', 1: 'spam'}
array = words_in_texts(words, train['email'])
df = pd.DataFrame({
    'facts': array[:,0],
    'article': array[:,1],
    'job' : array[:,2],
    'link':array[:,3],
    'hello':array[:,4],
    'type': train['spam'].map(d)
})
df = df.melt('type')
sns.barplot('variable','value','type',data=df)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9baf405278>

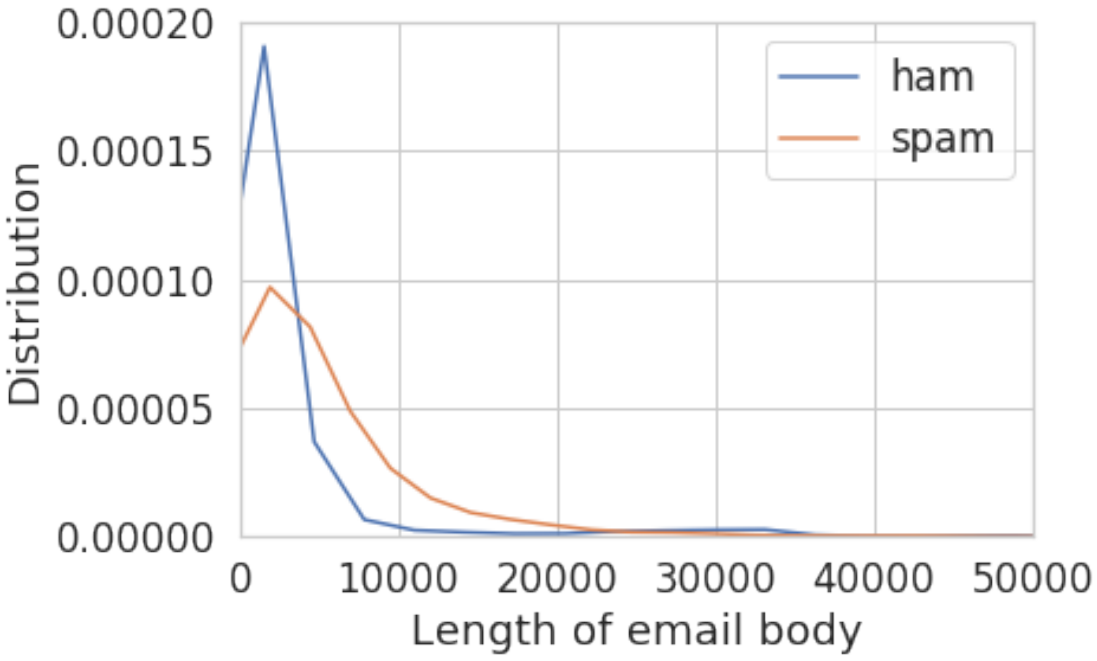


0.0.3 Question 3b

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]: df = pd.DataFrame({
             'email_length':train['email'].str.len(),
             'type': train['spam'].map(d)
         })
         sns.distplot(df[df['type']=='ham']['email_length'],hist=False,label='ham')
         sns.distplot(df[df['type']=='spam']['email_length'],hist=False,label='spam')
         plt.xlim(0, 50000)
         plt.ylabel('Distribution')
         plt.xlabel('Length of email body')
         plt.legend()
```

```
Out[14]: <matplotlib.legend.Legend at 0x7f9baf405048>
```



0.0.4 Question 6c

Provide brief explanations of the results from 6a and 6b. Why do we observe each of these values (FP, FN, accuracy, recall)?

Zero Classifier has no TP or FP and because of that the recall is zero. This gives us a standard we could compare to.

0.0.5 Question 6e

Are there more false positives or false negatives when using the logistic regression classifier from Question 5?

More False Positives

0.0.6 Question 6f

1. Our logistic regression classifier got 75.6% prediction accuracy (number of correct predictions / total). How does this compare with predicting 0 for every email?
2. Given the word features we gave you above, name one reason this classifier is performing poorly. Hint: Think about how prevalent these words are in the email set.
3. Which of these two classifiers would you prefer for a spam filter and why? Describe your reasoning and relate it to at least one of the evaluation metrics you have computed so far.

1. The accuracy for the zero-predictor was 74.47%.
2. The reason could be that there are very low number of ones, aka these words aren't prevalent in the email train set.
3. I think if we were to minimize false positives, then the zero_predictor works best. For a spam filter, we can't use these words to classify and if we were to use other words, I would prefer the logistic regression model because of better recall

0.0.7 Question 7: Feature/Model Selection Process

In this following cell, describe the process of improving your model. You should use at least 2-3 sentences each to address the follow questions:

1. How did you find better features for your model?
2. What did you try that worked / didn't work?
3. What was surprising in your search for good features?

1. First, I created a function for each and every suggestion provided and included those in the X_train. I also made word clouds using frequency datasets for words in spam and ham. Made the df for words in spam and ham using the counter from collections. Used the words apart from the first 25 or 50 to get words more likely to be in spam or ham emails. Was a lot of trial and error.

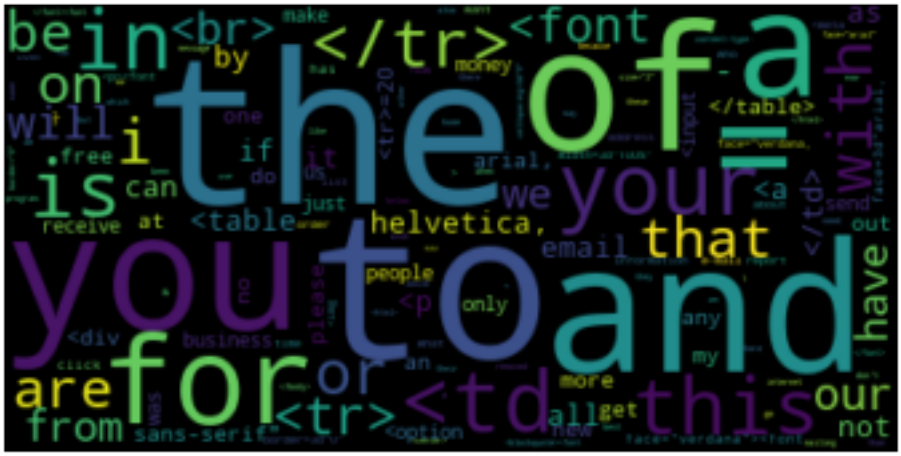
2. Tried different set of words.Tried without the functions and with the functions. Trial and error. Some words like font and content and 'a', didn't work whereas including words like fortune and ffffff and including the reply & forward function seemed to work.

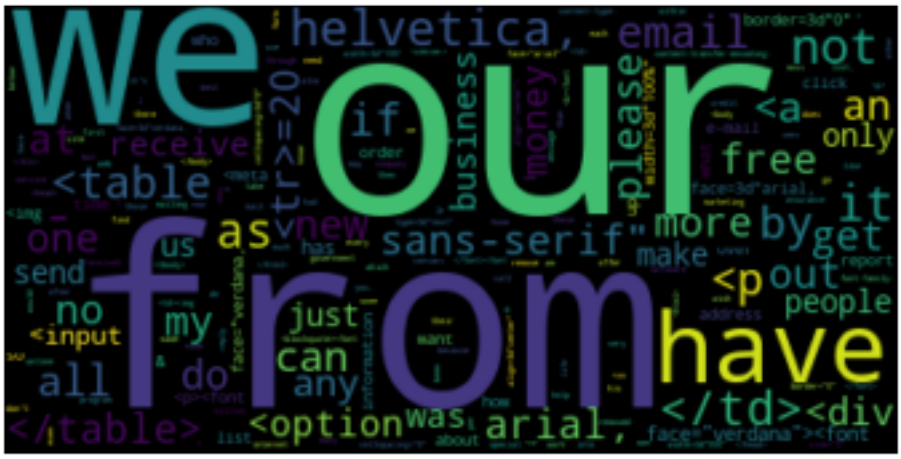
3. Some html tags, along with fonts didn't work at all. And most words that were more prevalent in spam than ham or opposite were parts of words like yo or in

Generate your visualization in the cell below and provide your description in a comment.

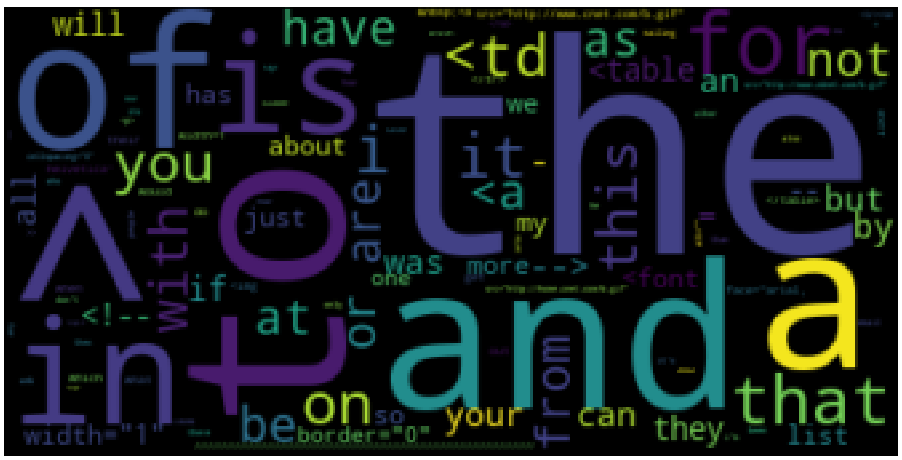
In [476]:
```
# Write your description (2-3 sentences) as a comment here:
# I used word clouds. Using word clouds of top words, top after 25, top after 50 gave me a se
# in the spam or ham emails. Using this and the visualization in 3 and trial and error, I cho
#

# Write the code to generate your visualization here:
d_1 = {}
for a, x in spam_df.values:
    d_1[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_1)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
d_2 = {}
for a, x in spam_df.values[25:]:
    d_2[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_2)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
d_3 = {}
for a, x in spam_df.values[50:]:
    d_3[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_3)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
d_4 = {}
for a, x in ham_df.values:
    d_4[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_4)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
d_5 = {}
for a, x in ham_df.values[25:]:
    d_5[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_5)
```

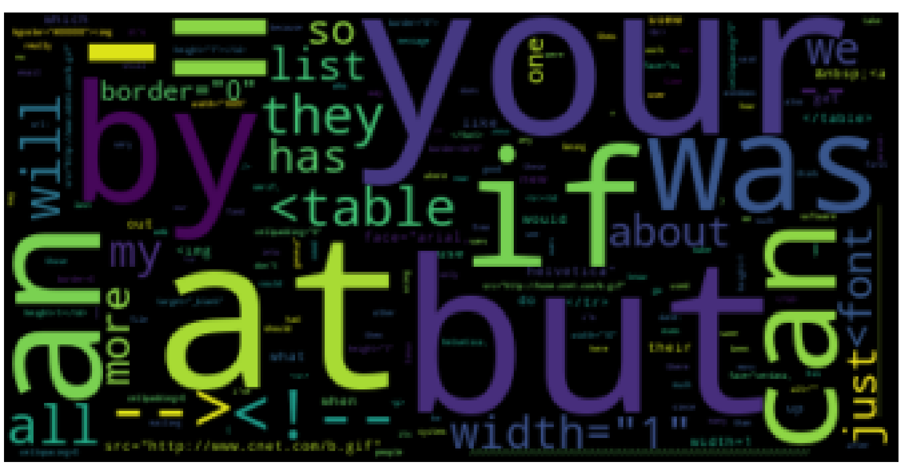
```
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
d_6 = {}
for a, x in ham_df.values[50:]:
    d_6[a] = x

wordcloud = WordCloud()
wordcloud.generate_from_frequencies(frequencies=d_6)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```






please
do
more
<tr>
=20
an
get
out
one
was
receive
just
business
information
<input
click
e-mail
how
us
what
address
make
money
send
any
only
about
new
<div
<option
people
list
face="verdana,
order
face=3d"arial,
border=3d"0"
<meta
face="verdana"><font
report
time
has
want
up
who
over
<p><font
width=3d"100%"
<img
so


of
is
the
for
to
and
a
in
>
have
will
<td
as
not
<table
an
has
we
you
about
i
it
-
<a
this
but
by
all
with
just
are
my
was
more-->
one
if
<!--
at
or
<font
face="arial,
from
on
be
so
your
can
they
that
list
width="1"
border="0"


your
by
=
so
list
we
border="0"
they
has
was
if
will
<table
about
an
my
at
but
can
more
<font
just
all
-->
<!--
width="1"
src="http://www.cnet.com/b.gif"
one
out
their


 <a
their
it's
only
other
new
how
use
mailing
face="arial,
helvetica,
people
<img
some
out
one
than
what
do
2002
there
alt=**
which
src="http://www.zdnet.com/b.gif"
up
don't
also
<tr
width=1
border="0"
anto
who
when
like
</table>
would
its
then
wrote:
cellspacing=
src="http://home.cnet.com/b.gif"
src="http://www.cnet.com/b.gif"
list
just

0.0.8 Question 9: ROC Curve

In most cases we won't be able to get no false positives and no false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover a disease until it's too late to treat, while a false positive means that a patient will probably have to take another screening.

Recall that logistic regression calculates the probability that an example belongs to a certain class. Then, to classify an example we say that an email is spam if our classifier gives it ≥ 0.5 probability of being spam. However, *we can adjust that cutoff*: we can say that an email is spam only if our classifier gives it ≥ 0.7 probability of being spam, for example. This is how we can trade off false positives and false negatives.

The ROC curve shows this trade off for each possible cutoff probability. In the cell below, plot an ROC curve for your final classifier (the one you use to make predictions for Kaggle). Refer to the Lecture 22 notebook or Section 17.7 of the course text to see how to plot an ROC curve.

In [477]:
```
from sklearn.metrics import roc_curve

# Note that you'll want to use the .predict_proba(...) method for your classifier
# instead of .predict(...) so you get probabilities, not classes
fpr, tpr, thresholds = roc_curve(Y_test, log_reg.predict_proba(X_val_mod)[:,1])
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Test Data');
```