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.

A noticable difference is that the spam email uses HTML while ham is just text.

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`.

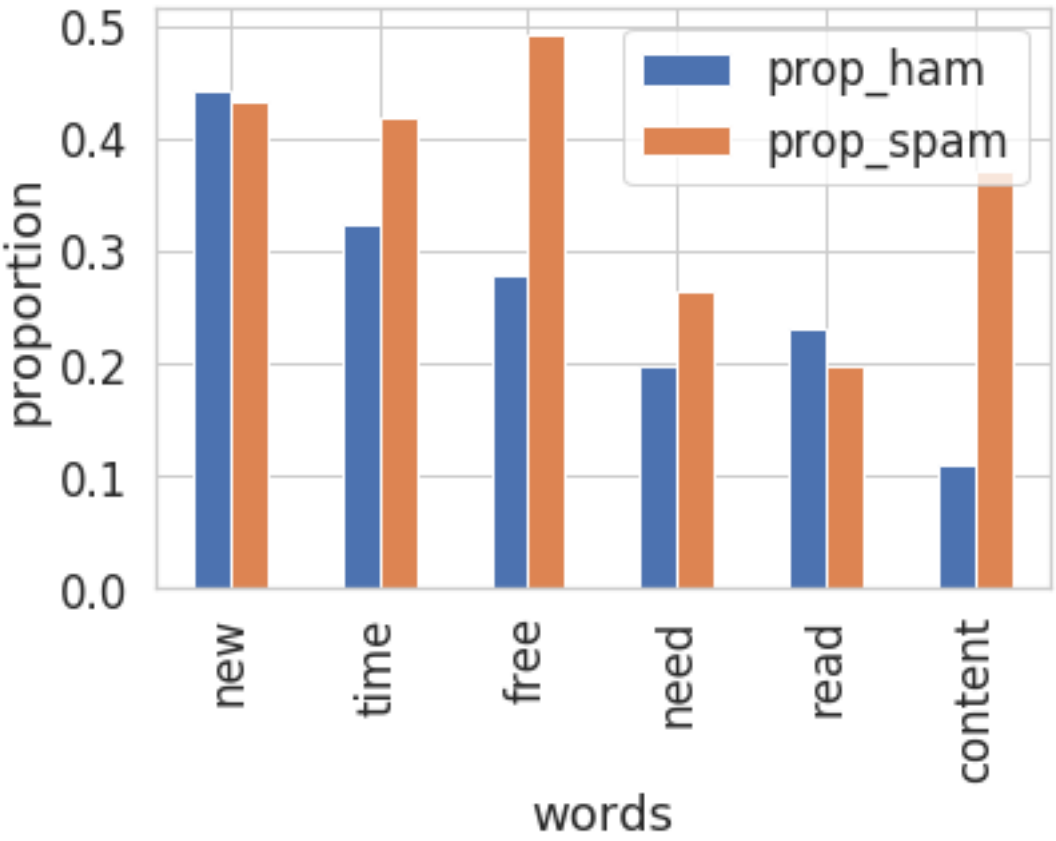
In [13]:
```
words = ['new', 'you', 'free', 'need', 'read', 'content']
words_table = pd.DataFrame(words_in_texts(words, train['email']), columns = words)
words_table['type'] = train['spam']
words_table.head(5)
```

Out[13]:

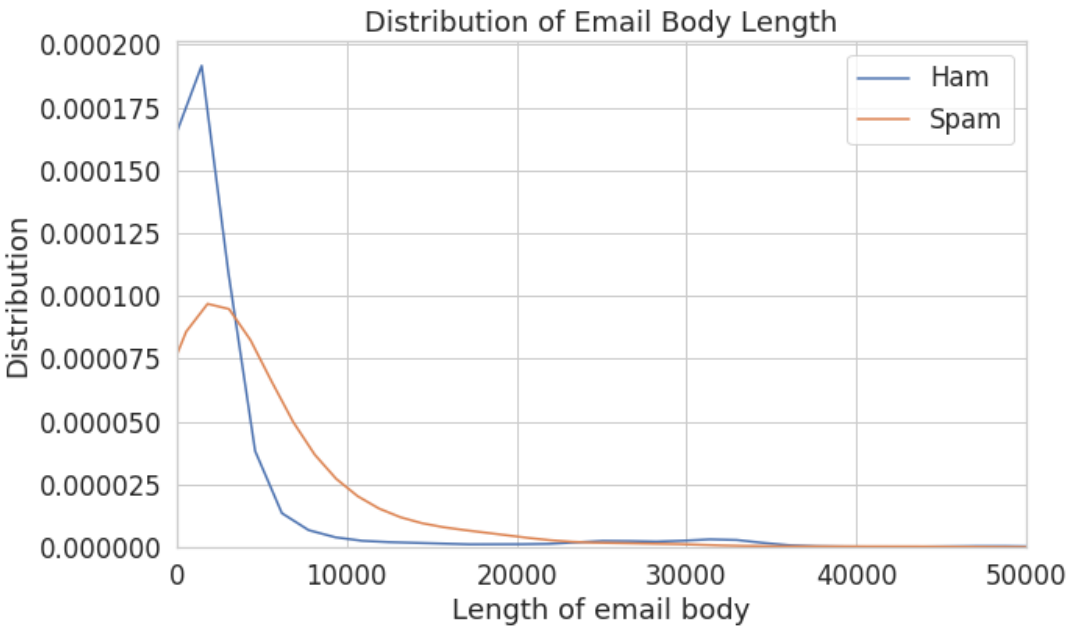
	new	you	free	need	read	content	type
0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
1	1.0	1.0	0.0	0.0	1.0	0.0	0.0
2	1.0	1.0	1.0	0.0	0.0	0.0	1.0
3	1.0	1.0	0.0	0.0	1.0	0.0	0.0
4	1.0	1.0	1.0	1.0	1.0	0.0	0.0

In [14]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai
words = ['new', 'time', 'free', 'need', 'read', 'content']
words_table = pd.DataFrame(words_in_texts(words, train['email']), columns = words)

prop_ham = []
prop_spam = []

for i in words:
    type_ham = len(train[(train['spam'] == 0) & (words_table[i] == 1)]) / len(train[train['spa
    prop_ham.append(type_ham)
    type_spam = len(train[(train['spam'] == 1) & (words_table[i] == 1)]) / len(train[train['sp
    prop_spam.append(type_spam)

df = pd.DataFrame(data = {'prop_ham':prop_ham, 'prop_spam':prop_spam})
df['words'] = words
df = df.set_index('words')
df = df.plot.bar()
df.set(xlabel = 'words', ylabel='proportion')
```

Out[14]: [Text(0, 0.5, 'proportion'), Text(0.5, 0, 'words')]

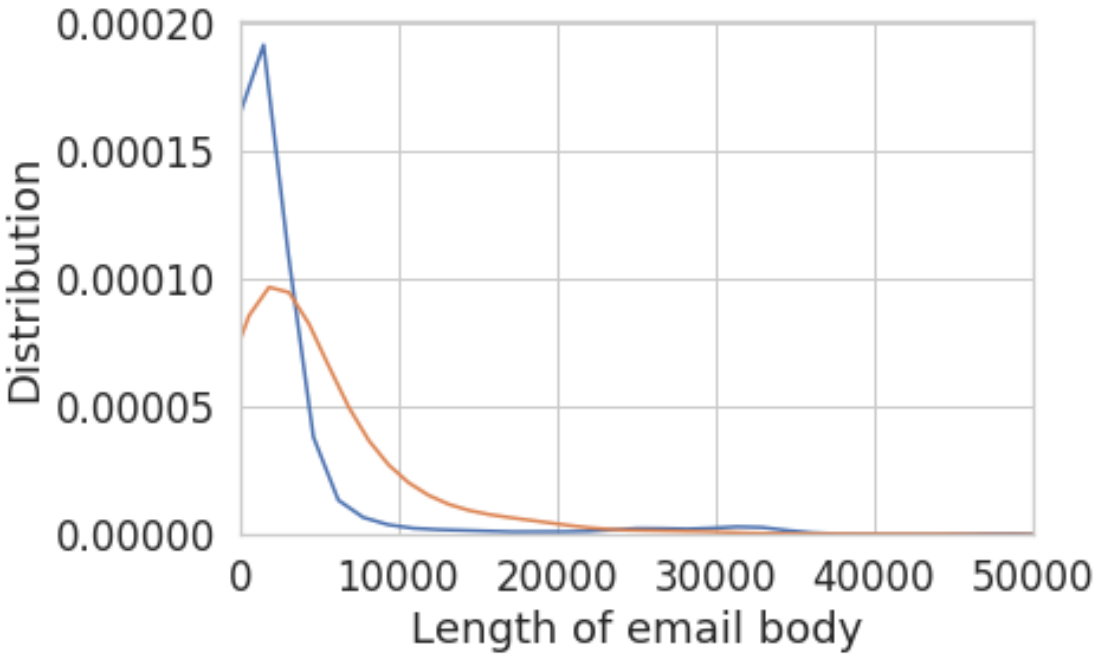

0.5
0.4
0.3
0.2
0.1
0.0
proportion
prop_ham
prop_spam
new
time
free
need
read
content
words

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 [15]: sns.distplot(train[train['spam']==0]['email'].str.len(), hist=False, label='ham')
         sns.distplot(train[train['spam']==1]['email'].str.len(), hist=False, label='spam')

         plt.xlabel('Length of email body')
         plt.ylabel('Distribution')
         plt.xlim(0, 50000)
         plt.savefig('training_conditional_densities.png')
```


0.00020
0.00015
0.00010
0.00005
0.00000
Distribution
0
10000
20000
30000
40000
50000
Length of email body

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)?

We observe these values because, FP will always be zero because zero_predictor always predicts zero. FN indicates the emails that are being mislabeled as spam instead of ham. Accuracy of zero_predictor represents the proportion of ham emails that are labeled ham. Recall is represents the TP which will just be 0.

0.0.5 Question 6e

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

There are more false positives when using the logistic regression classifier from Question 5.

0.0.6 Question 6f

1. Our logistic regression classifier got 75.76% 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. Comparing the logistic regression classifer and the zero_predictor, the performance of the logistic regression classifier is poor because it's accuracy is very close to the zero_predictor.
2. Drug is probably not prevalent therefore, it is not a good feature resulting in the classifier performing poorly.
3. Of the two I would prefer the logistic regression classifier because it's accuracy, although slightly, is still higher than the zero_predictor.

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 or didn't work?
3. What was surprising in your search for good features?

1) I first split the training set and extracted the text from the html. Then I made all the text into two lists one for ham and other for spam. Using these two lists I created a dictionary that would contain the word and the number of times it appeared. I sorted the dictionary so that it would be in ascending order and got the 500 most common words. Lastly, I got the words that only applied to spam.

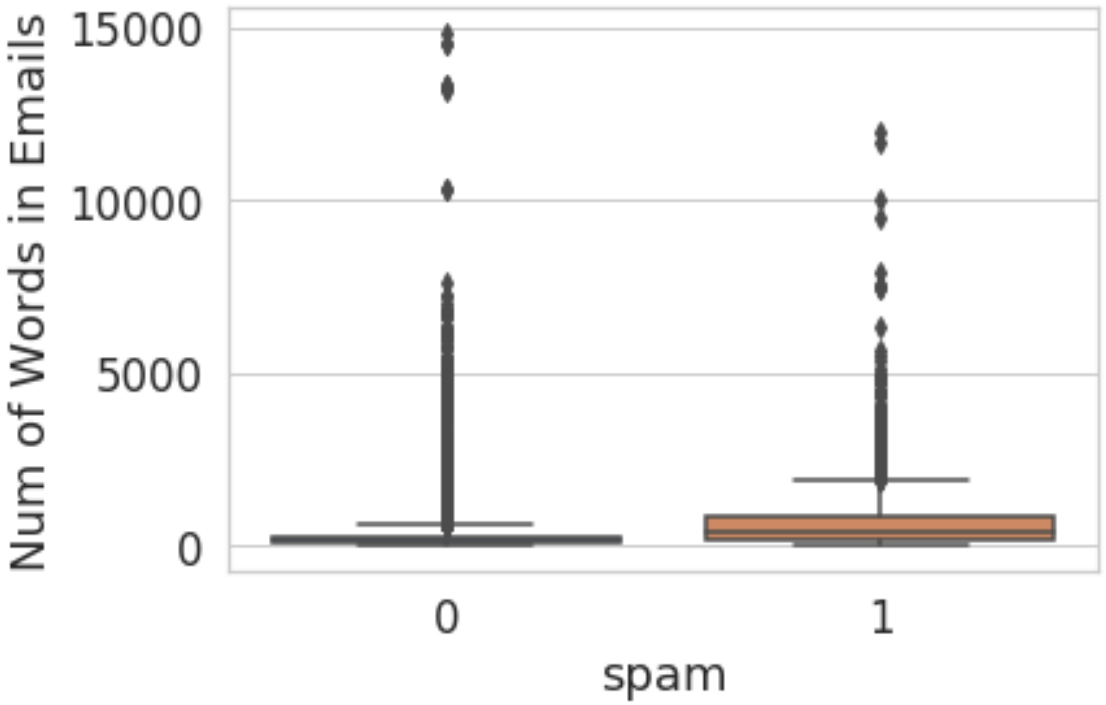
2) I tried to see whether the subject of ham or spam emails had more words. After plotting this feature, there was no obvious difference in the number of words, therefore it did not seem that useful as a feature. Another thing that I tried was seeing the use of capital letters in the subjects. Spam emails had more capitals in the subjects, however, the difference did not seem to be enough to distinguish between ham and spam emails. Lastly, I tried to use emails that were replies and saw that only ham emails had replies while spam had none. I ended up choosing to focus on words that were unique to spam emails, therefore, this feature was unnecessary but would have been a good feature to use if I chose different features.

3) What was suprising was that many of the text that were used as features in my models were not even words. From the 100 more common words that were only in the spam emails a good portion were just numbers or a combination of numbers of letters. Furthermore, while it was somewhat expected, it was still suprising to see that many of the words were money-related, such as price, quote, insurance, financial, card and etc.

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

In [35]:
```
# Write your description (2-3 sentences) as a comment here:
# This plot shows the total number of words in ham and spam emails.
# Even though this does not show the individual lengths of emails, there were around 4000 more
# This clearly shows that spam emails are wordier in general.
# Used words from spam emails as features because there are less spam emails but more words fr
# to differentiate between the emails.

# Write the code to generate your visualization here:
words_num_email = train['email'].str.replace(r'[^0-9a-zA-Z]', ' ').str.split().apply(lambda x:l
train['Num of Words in Emails'] = words_num_email
sns.boxplot(x ='spam', y= 'Num of Words in Emails', data= train)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6c4550b160>
```



0.0.8 Question 9: ROC Curve

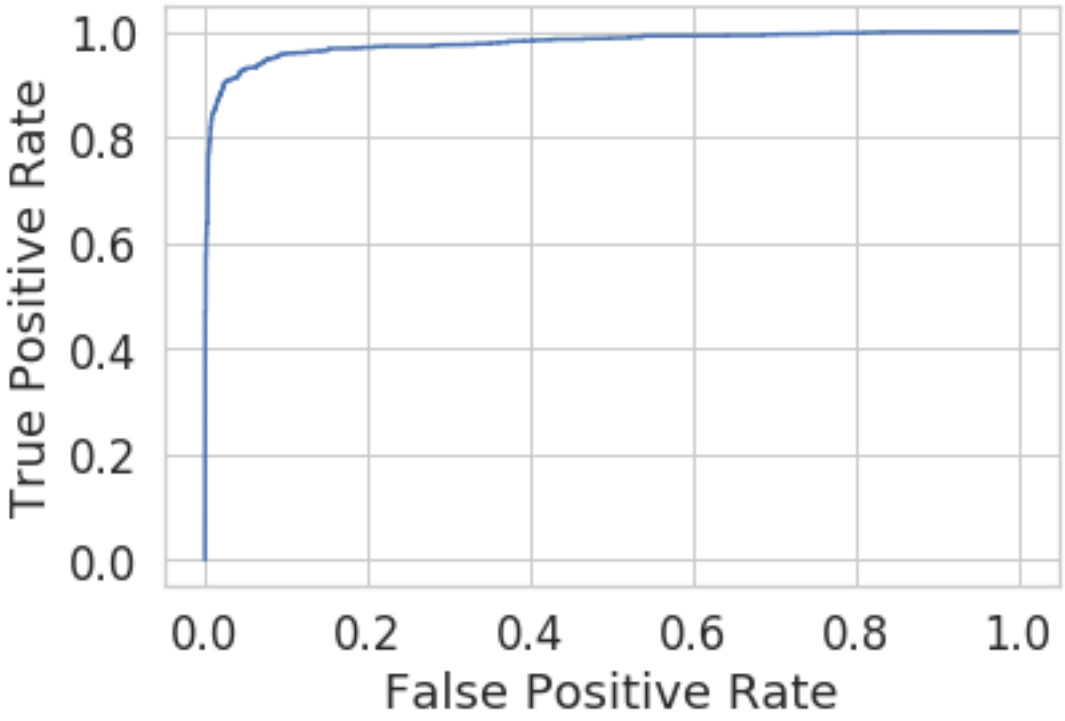
In most cases we won't be able to get 0 false positives and 0 false negatives, so we have to compromise. For example, in the case of cancer screenings, false negatives are comparatively worse than false positives — a false negative means that a patient might not discover that they have cancer until it's too late, whereas a patient can just receive another screening for a false positive.

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 a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 19 or Section 17.7 of the course text to see how to plot an ROC curve.

In [36]:
```
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

my_model_probabilites = my_model.predict_proba(X_train_part_2)[:,1]

fpr, tpr, thresholds = roc_curve(Y_train_part_2, my_model_probabilites,)

plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[36]:
```
Text(0, 0.5, 'True Positive Rate')
```


True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate