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 spam email seems to have links to raw IP addresses and is HTML-formatted. The ham email has links to actual dynamically-DNS'd sites and appears to be formatted in some form of Markdown.

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 [12]:
```
words = (
    train["email"]
    .str.replace("[^\w\s]", "")
    .str.split(" ").explode()
    .str.split("\\").explode()
    .value_counts()
    .filter(regex="[\w]")
)
```

In [13]:
```
test_words = ["linux", "enlargement", "enlarge",
              "opportunity", "pill", "pills", "insight",
              "minix", "invest", "investment", "kernel",
              "science"]
for w in test_words:
    try:
        print(f"{w}:\t{words[w]} occurences")
    except:
        print(f"{w}:\t0 occurences")
```

```
linux:        1906 occurences
enlargement:        12 occurences
enlarge:       17 occurences
opportunity:        439 occurences
pill:       44 occurences
pills:       29 occurences
insight:       31 occurences
minix:       0 occurences
invest:       143 occurences
investment:        476 occurences
kernel:       421 occurences
science:       148 occurences
```

In [14]:
```
train=train.reset_index(drop=True)
# We must do this in order to preserve the ordering of emails to labels for words_in_texts

words = ["enlargement", "enlarge",
         "pill", "investment",
         "kernel", "linux", "science"]
eda_word_freq = (
    pd.DataFrame(words_in_texts(words, train["email"]), columns=words)
```

```
        .merge(train["spam"], left_index=True, right_index=True)
        .melt("spam")
    )
```

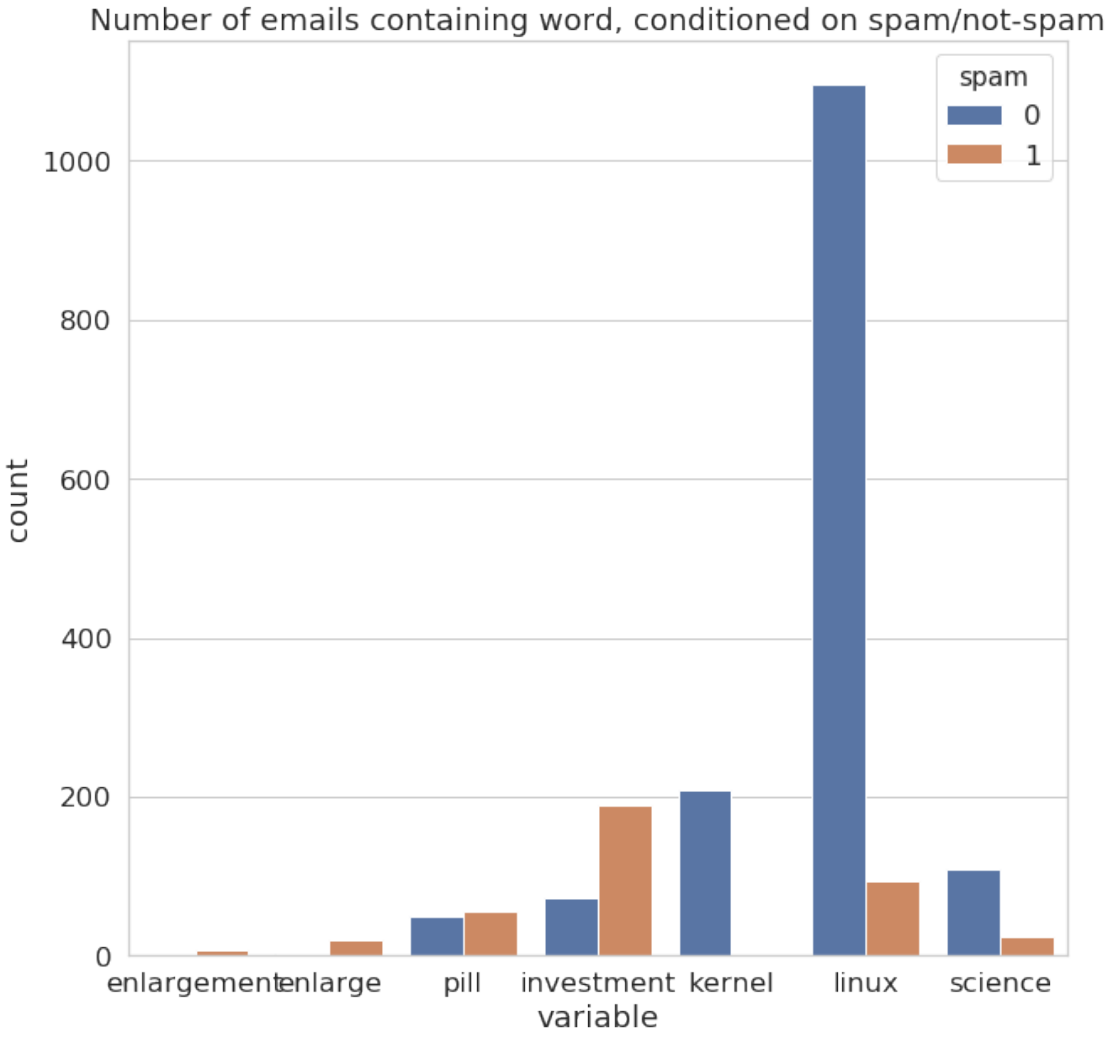
In [15]:
```
eda_word_freq
```

Out[15]:

	spam	variable	value
0	0	enlargement	0
1	0	enlargement	0
2	0	enlargement	0
3	0	enlargement	0
4	0	enlargement	0
…	…	…	…
52586	0	science	0
52587	1	science	0
52588	0	science	0
52589	0	science	0
52590	0	science	0

[52591 rows x 3 columns]

In [16]:
```
plt.figure(figsize=(10, 10))
sns.countplot(
    x="variable",
    hue="spam",
    data=eda_word_freq[eda_word_freq["value"] == True]
)
plt.title("Number of emails containing word, conditioned on spam/not-spam")
plt.xlabel = "word"
```

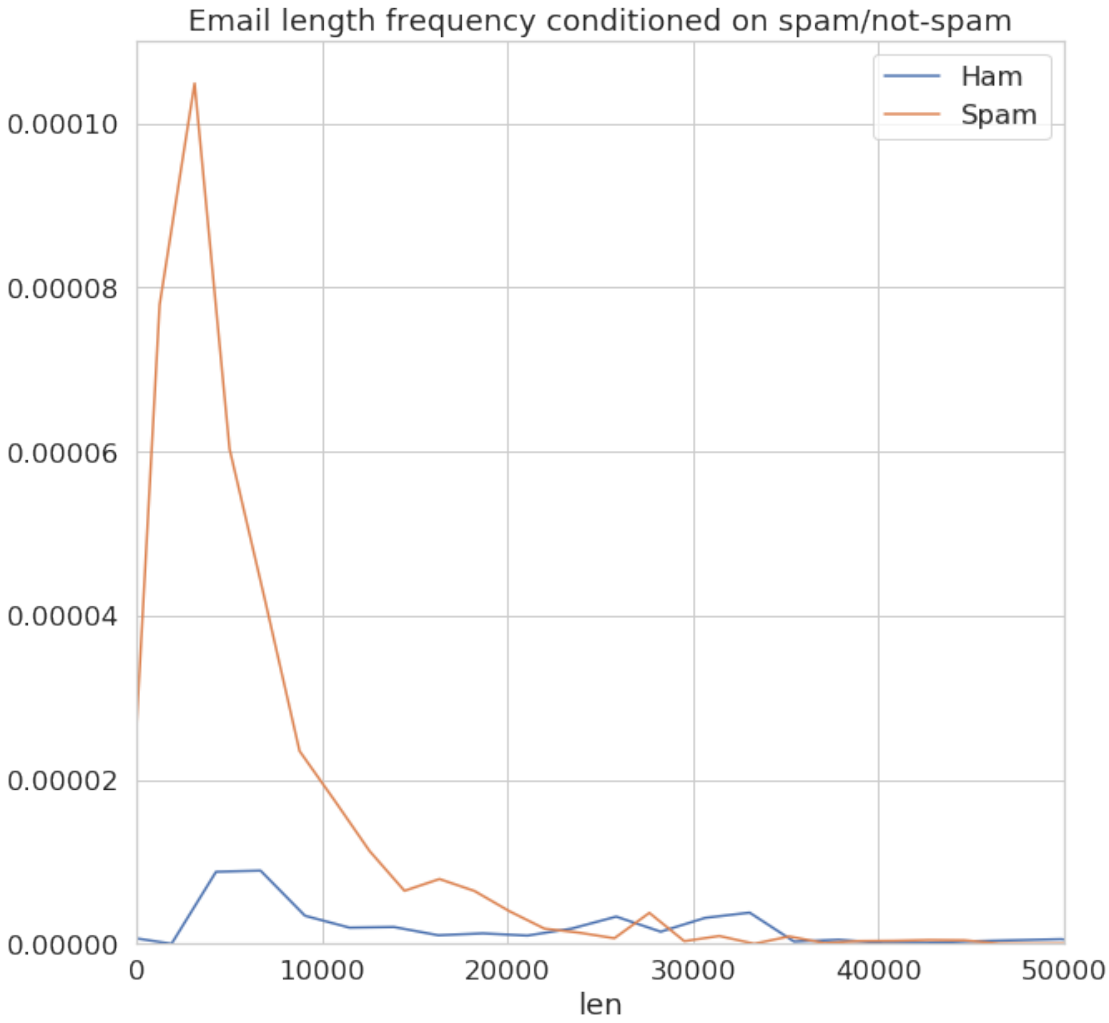

Number of emails containing word, conditioned on spam/not-spam
spam
0
1
count
1000
800
600
400
200
0
enlargement
enlarge
pill
investment
kernel
linux
science
variable

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 [17]:
```
train["len"] = train["email"].str.len()
```

In [18]:
```
plt.figure(figsize=(10, 10))
sns.distplot(
    train[train["spam"] == 0]["len"],
    label="Ham",
    hist=False
)
sns.distplot(
    train[train["spam"] == 1]["len"],
    label="Spam",
    hist=False
)
plt.legend()
plt.xlim((0, 50000))
plt.title("Email length frequency conditioned on spam/not-spam")
plt.xlabel="email length"
plt.ylabel="frequency"
```


Email length frequency conditioned on spam/not-spam
Ham
Spam
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
10000
20000
30000
40000
50000
len

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 have a very low false positive rate because we *aren't marking any emails as positive,* which means that we don't identify *any* spam emails. For the same reason, we have very low recall: we aren't correctly identifying any of the spam emails. It also means that our accuracy is equal to the proportion of non-spam emails, since we only correctly identify the non-spam emails. However, our false negative rate is equal to the number of spam emails, since we have failed to identify *any* of the spam emails.

0.0.5 Question 6e

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

There are far more false negatives than 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. This is slightly better than predicting zero for every email, which got just over 74% accuracy.

2. The classifier is performing poorly because the words we've trained it on are not *overwhelmingly* indicative of spam, or of ham. For instance, the word "prescription" is not necessarily spam, as it could be a legitimate doctor's email, but neither is it ham, as it could be some homeopathic remedy.

3. I would prefer the zero predictor. This is because it doesn't misclassify any ham emails as spam, whereas the logistic regression would have moved 122 ham emails to the spam folder (type 2 error) which is suboptimal.

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. I found better features by checking the validation accuracy and recall on logistic regressions trained on exclusively that feature. I picked the ones which had the highest individual accuracy, and then from these, I selected the features that were the least redundant (from a heuristic standpoint) and that had the largest negative effect on the model bias.
2. I was surprised to find that the most important features were words, ie. the semantic content of the messages. I started out with a larger model incorporating data such as the proportion of capitalised letters in the message or subject, and the number of HTML tags, etc. but after excluding these features, I found that they didn't actually have a very large effect on the accuracy or recall of the model.
3. About half of the spam emails could be filtered out just using a vocabulary filter, but the remaining half proved far more difficult to filter. Ideally, a higher-level semantic filter could be applied to the actual email body to further analyse the meaning of the email.

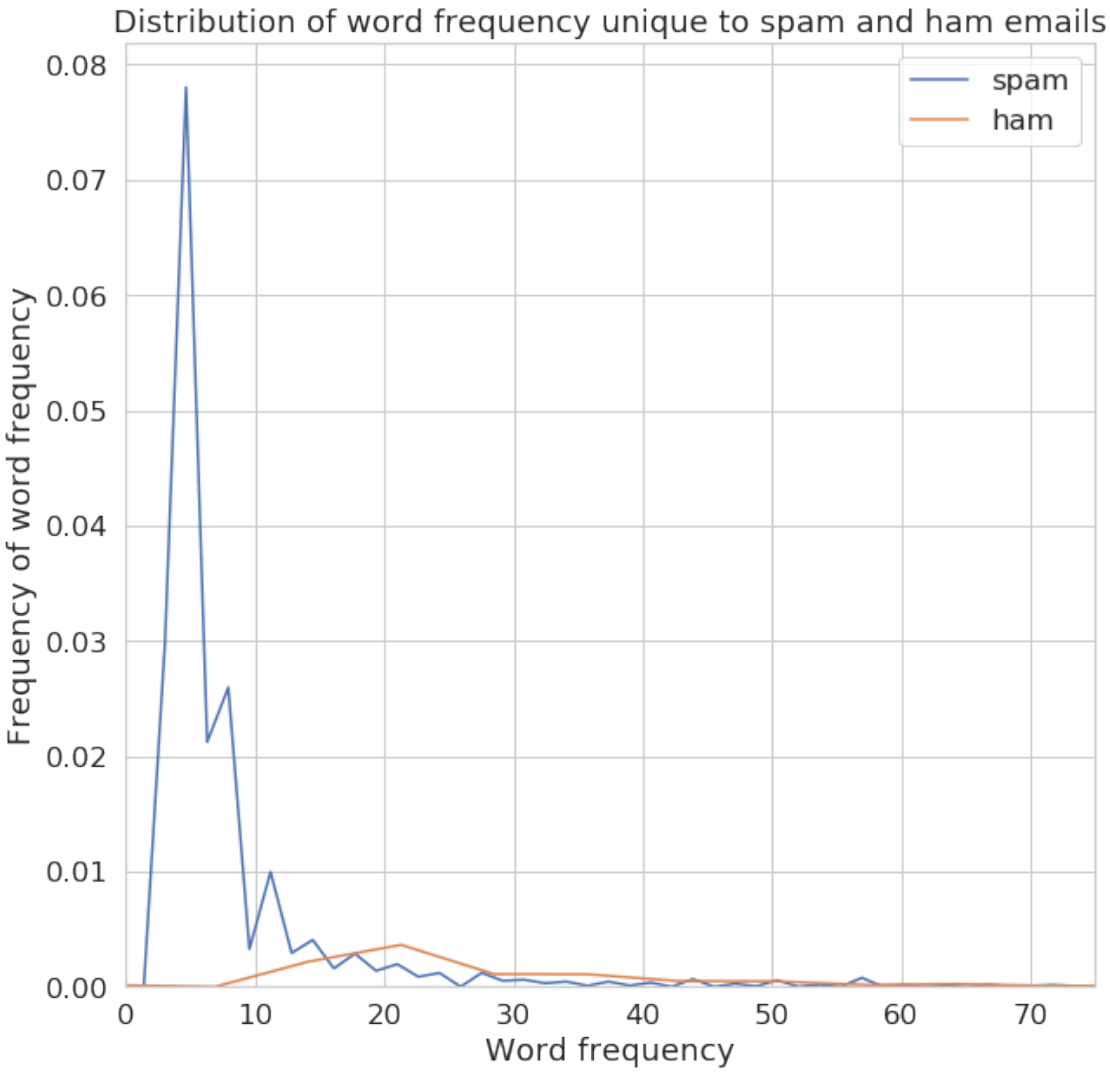
Generate your visualization in the cell below and provide your description in a comment.

In [68]:
```
# Write your description (2-3 sentences) as a comment here:
#
# I decided to take a look at how the most common words exclusive
# to spam emails were distributed, in comparison to the most common
# words exclusive to ham emails. It turns out that the most common
# spam-exclusive words occur far more often than the most common
# ham-exclusive words, This could be because spam emails tend
# to have a laser-sharp focus on certain subjects such as finance
# scams or genital size enhancements, while legitimate ham emails
# are far more varied in their content.

# Write the code to generate your visualization here:

fig = plt.figure(figsize=(10, 10))
sns.kdeplot(unique_spam_words.value_counts(), label="spam")
(sns.kdeplot(unique_ham_words.value_counts(), label="ham")
    .set(
        title="Distribution of word frequency unique to spam and ham emails",
        xlabel="Word frequency",
        ylabel="Frequency of word frequency"
    )
)
plt.xlim(0, 75)
plt.legend()
```

Out[68]:
```
<matplotlib.legend.Legend at 0x7f5c0634d250>
```


Distribution of word frequency unique to spam and ham emails
spam
ham
Frequency of word frequency
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
60
70
Word frequency

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.

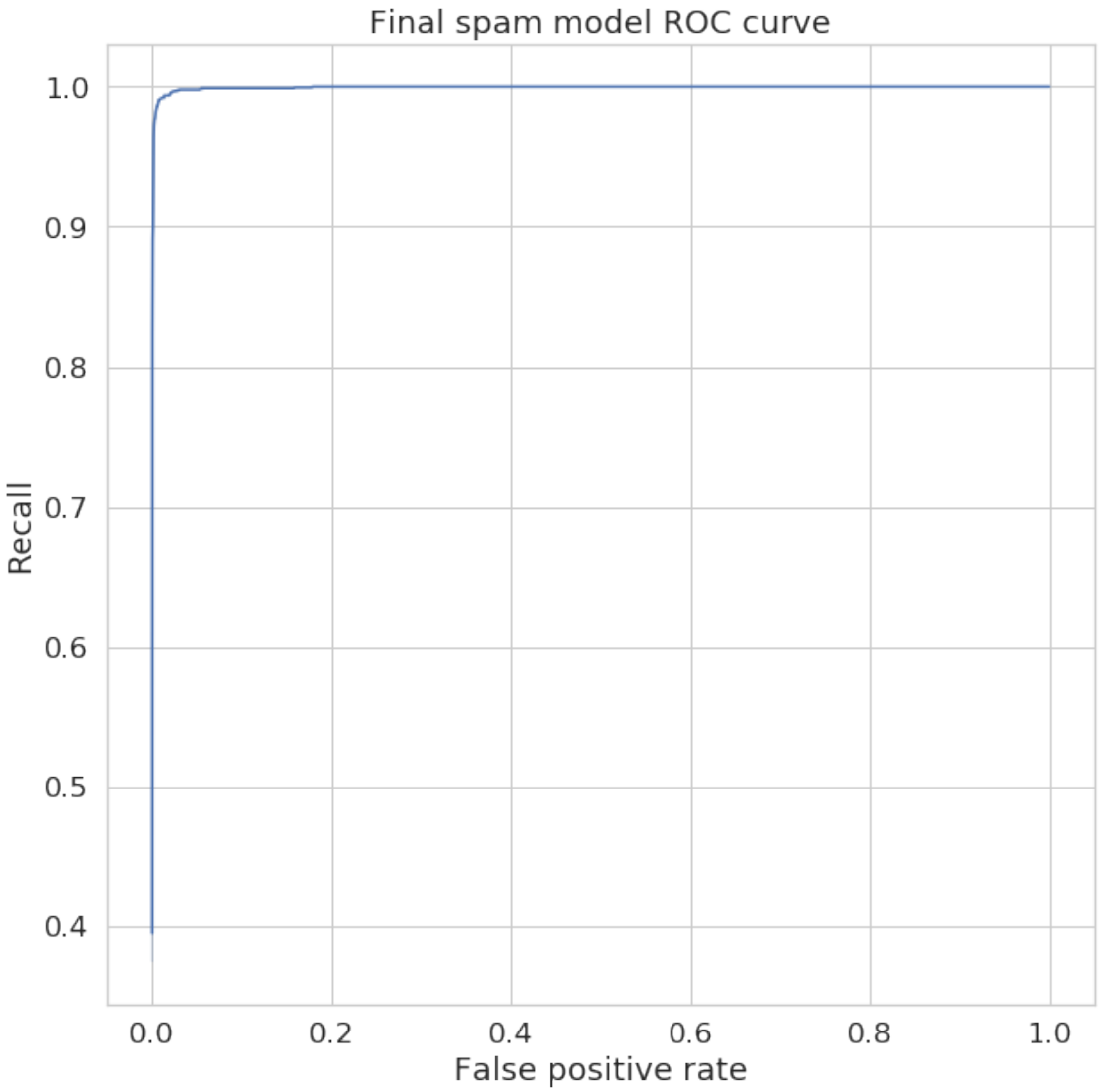
The ROC curve shows this trade off for each possible cutoff probability. In the cell below, plot a ROC curve for your final classifier (the one you use to make predictions for Gradescope) on the training data. Refer to Lecture 19 or Section 17.7 of the course text to see how to plot an ROC curve.

In [69]:
```
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

probabilities = final_model.predict_proba(X)[:, 1]
false_positive_rate_values, recall_values, thresholds = roc_curve(y, probabilities, pos_label=
plt.figure(figsize=(10, 10))
(sns.lineplot(x=false_positive_rate_values, y=recall_values)
    .set(
        xlabel="False positive rate",
        ylabel="Recall",
        title="Final spam model ROC curve"
    )
)
```

Out[69]:
```
[Text(0, 0.5, 'Recall'),
 Text(0.5, 0, 'False positive rate'),
 Text(0.5, 1.0, 'Final spam model ROC curve')]
```


Final spam model ROC curve
Recall
False positive rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0