0.0.1 Question 1c

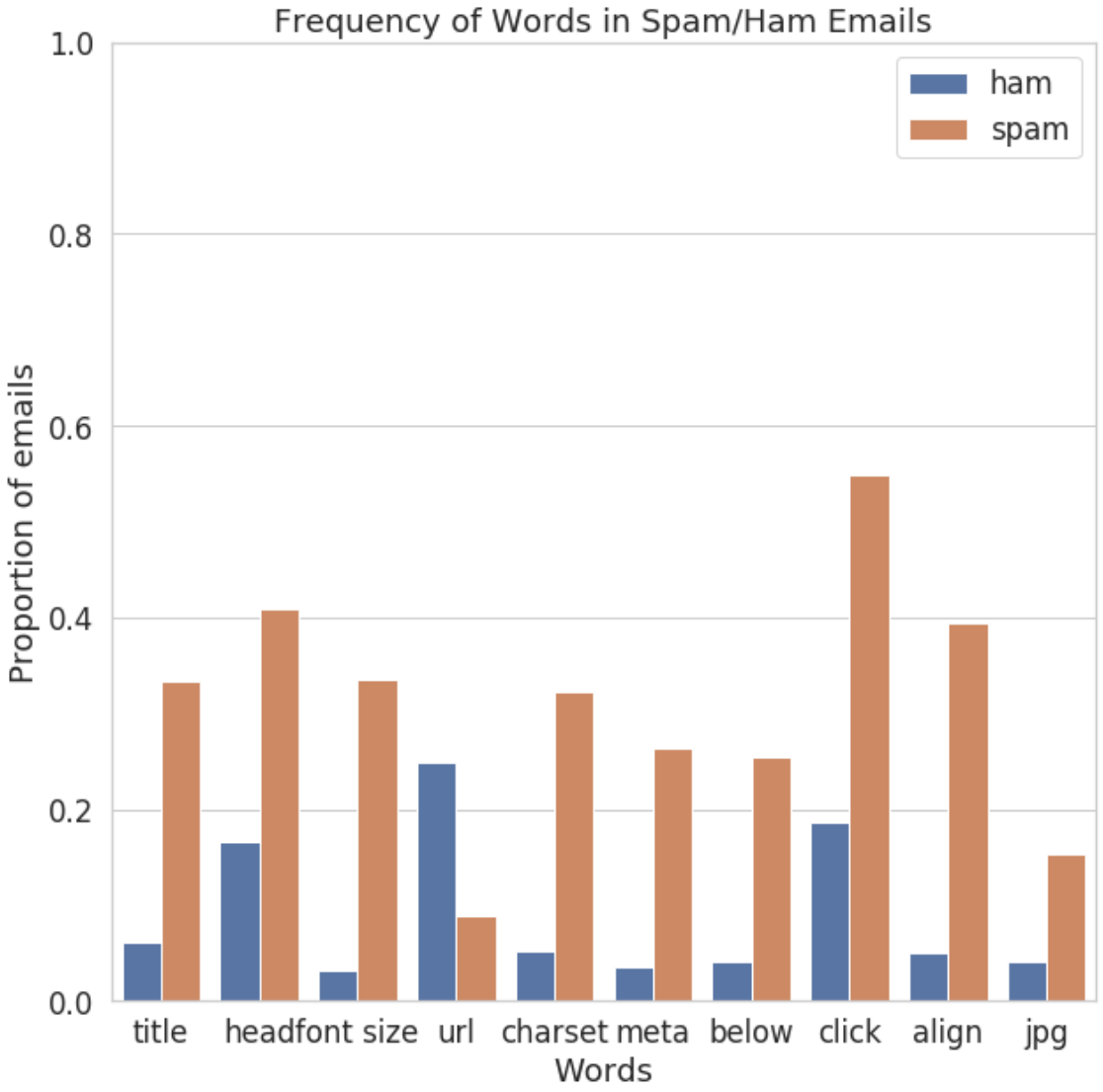
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

It seems like the spam email is formatted as a HTML format, while the ham is not. Also, there are new blank lines in the ham, while the spam is compacted with no new blank lines.

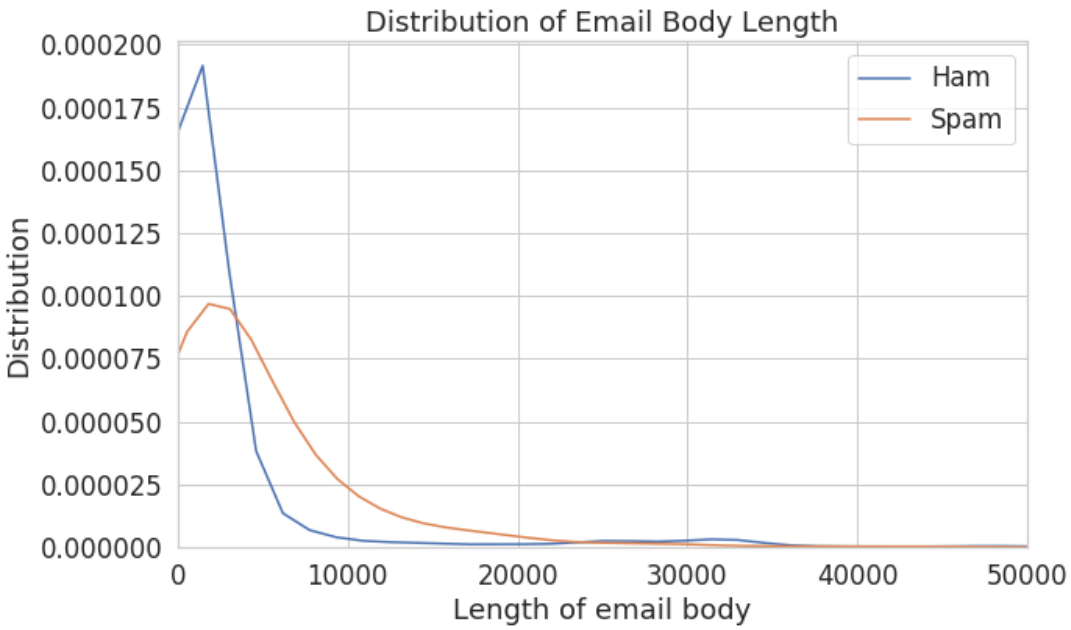
0.0.2 Question 3a

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 [107]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of ema
words = ['title', 'head', 'font size', 'url', 'charset', 'meta', 'below', 'click', 'align', '
texts = train['email']
new_table = pd.DataFrame(words_in_texts(words, texts))
new_table.columns = words
new_table["spam"] = train['spam']
new_table = new_table.melt("spam")
new_table['spam'] = new_table['spam'].replace({0: "ham", 1: "spam"})
plt.figure(figsize=(10, 10))
plt.ylim(0, 1)
ax = sns.barplot(x = 'variable', y = 'value', hue = 'spam', data=new_table, ci = None)
ax.legend_.set_title(None)
ax.set(xlabel='Words', ylabel='Proportion of emails')
ax.set_title("Frequency of Words in Spam/Ham Emails")
plt.savefig('myversionfrequnency.png');
```

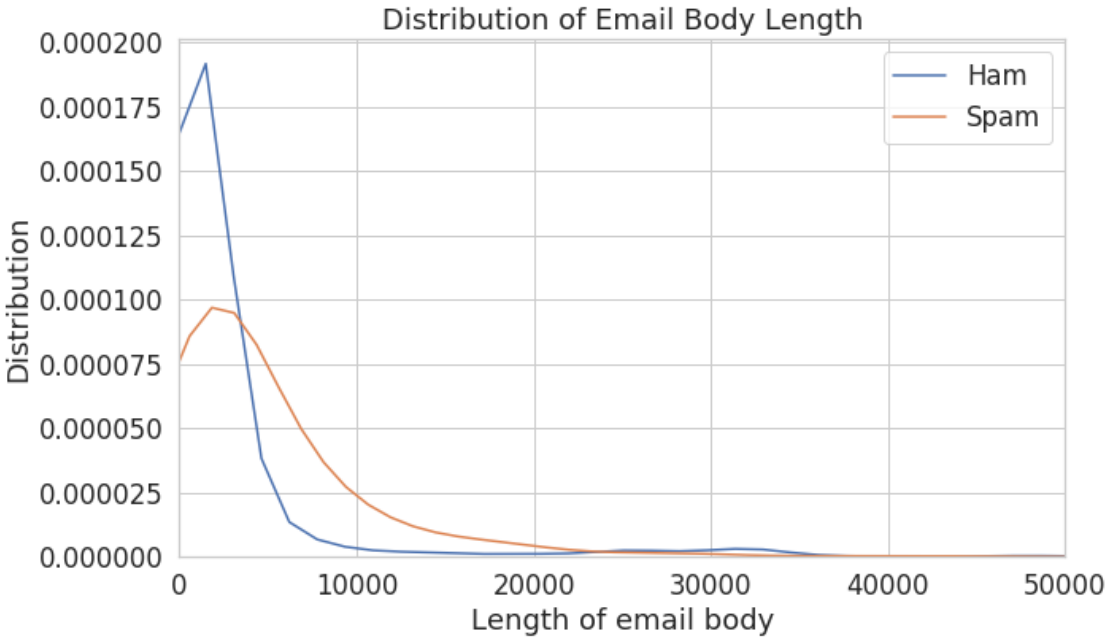

Frequency of Words in Spam/Ham Emails
ham
spam
Proportion of emails
1.0
0.8
0.6
0.4
0.2
0.0
title
headfont size
url
charset meta
below
click
align
jpg
Words

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 [108]: train['length'] = train['email'].str.len().astype(float)
          train['type'] = new_table['spam'].replace({0: "ham", 1: "spam"})
          spam = train[train['type'] == 'spam']
          ham = train[train['type'] == 'ham']
          plt.figure(figsize=(10, 6))
          spamhamdens = sns.distplot(ham, x = ham['length'], hist = False).set_xlim(0, 50000)
          sns.distplot(spam, x = spam['length'], hist = False)
          plt.xlim(0, 50000)
          plt.title('Distribution of Email Body Length')
          plt.xlabel('Length of email body')
          plt.ylabel('Distribution')
          plt.legend(labels = ['Ham', 'Spam']);
          plt.savefig('training_conditional_densities.png')
```


Distribution of Email Body Length
Ham
Spam
Distribution
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
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 calculate FP because we want to know how many we predict 1(spam) but in fact a 0(ham). Fn for knowing how many we predict 0(ham) but in fact a 1(spam). Accuracy for what proportion of points did our zero predictor classify correctly. However, accuracy doesn't tell the full story, so we need recall to get to know of all the emails that were actually spam(1), what proportion were actually spam(1) to get a better gasp on our zero predictor.

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 positive and less false negative when using the logistic regression classifier.

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.) It increases around 1% of the accuracy if we use the logistic regression. Though the difference is not huge, by using the logistic regression classifier, we are also predicting both of the classes, not just one like what the zero predictor did which it just predicts the 0 class. By using the logistic regression, it gives us a full picture on the predictions, like how well we are predicting spams within the emails, and how well we are predicting hams among the emails.

2.) Business, money, and offer seem to be also very prevalent in ham emails, thus making this classifier performing poorly.

3.) I would prefer the logistic regression classifier because it give a full story especially in cases with high class inbalance. The zero predictor is only predicting on ham emails, but it is not predicting well in the spam emails. As we can see, the recall is 0 for the zero predictor, which of all the observations that were actually 1, we did poorly on predicting howmany of them are actually 1(spam). Even though the recall when using the logistic is still not as good, but at least it is better than the zero predictor's.

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 used the tendency of most spam emails in the format of html and adding words that are specifically in html tags(like head, font size, charset, meta) into the word list. But later I found out it is not enough to reach at least 88%, so I keep adding words that I observed when surfing throught the emails, and keep adding word to the list. It helped a bit and enough to reach 88%, however, I want to further improve it as much as possible to 100%. So instead of using my bare eyes to get a list of words, I try to implement a function to obtain a list of words with their counts. It turns out it improved my model a lot better, and I keep adding the amount of words I found through the function to my model and finally reached an accuracy of 100 in both my training data and through my cross validation process. Also, I try removing html tags and extract the words, turns out my accuracy got better, and it is my best accuracy out of all.

2.) At first I keep adding new features without really looking into them on whether they really define clearly emails to spam or ham. As I went back and look into the features I have added and created graphs to visualize them, I realize one of them, the counting of exclamation marks in emails, doesn't define well between the two classes, so I have decided to remove it, and the accuracy got better. It was the redundancy of data, like instead of adding a column for a count for a string, I found out addinf that string to my word list is actually better, so I removed as much as possible of those redundancies. I also found that getting a list of words manually is not enough to improve the accuracy, so at last I used a function I have created to help me get a better list of words.

3.) I am pretty surprised that punctuations are actually pretty good features, my accuracies kept increasing as I kept adding them into my word list. Also, as I am suspecting a lot of features that could mostly be found in spam emails, such as html tags, it turns out they could be found more often in ham emails, and I was surprised to find out that.

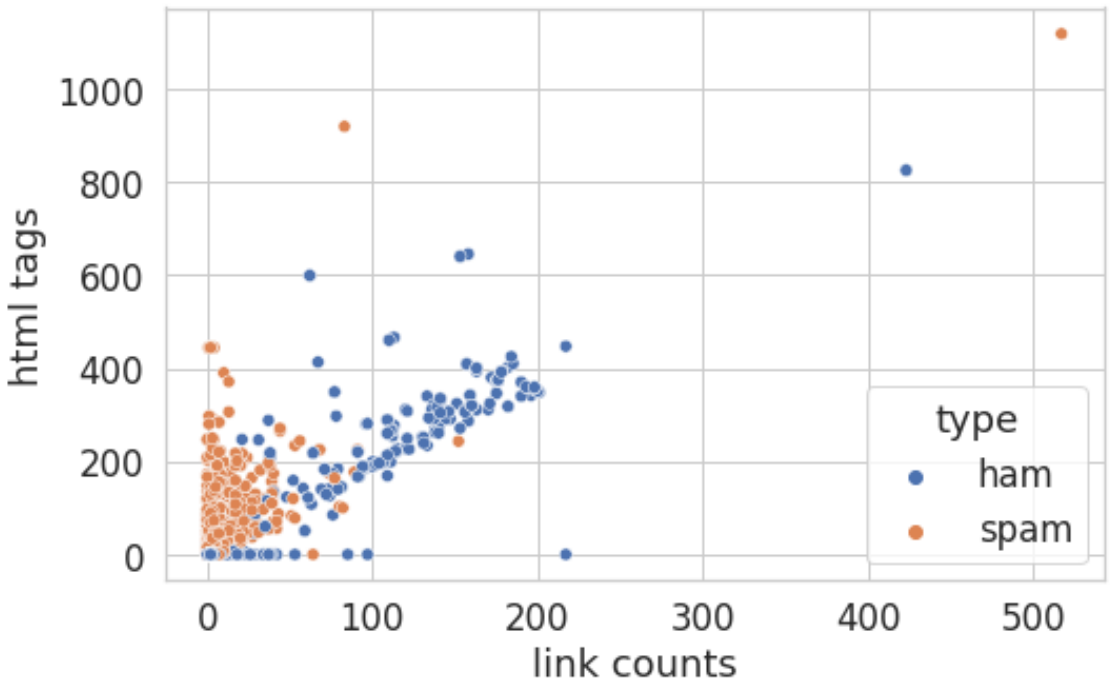
Generate your visualization in the cell below and provide your description in a comment.

In [173]:
```
# Write your description (2-3 sentences) as a comment here:
# In the graph below, it looks like there are some correlation between links counts and html
# The correlation is strong for some ham emails with some outliers
#In the case for spam emails, there might be some correlation but it is not as strong as for

# Write the code to generate your visualization here:

top_10_spam = list(spam_dict.keys())[1:11]
top_10_ham = list(ham_dict.keys())[1:11]
top_10_spam[0] = "next line"
top_10_ham[0] = "next line"
spam_type = ['spam']*10
spam_d = {'words': top_10_spam, 'count': list(spam_dict.values())[1:11], 'type' : spam_type}
spam_words_data = pd.DataFrame(data = spam_d)
ham_type = ['ham']*10
ham_d = {'words': top_10_ham, 'count': list(ham_dict.values())[1:11], 'type' : ham_type}
ham_words_data = pd.DataFrame(data = ham_d)
word_data= pd.concat([ham_words_data, spam_words_data])
plt.figure(figsize=(8, 5))
sns.scatterplot(x = "link counts", y = "html tags", data = train_copy, hue = "type")
```

Out[173]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd89911ae80>



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.

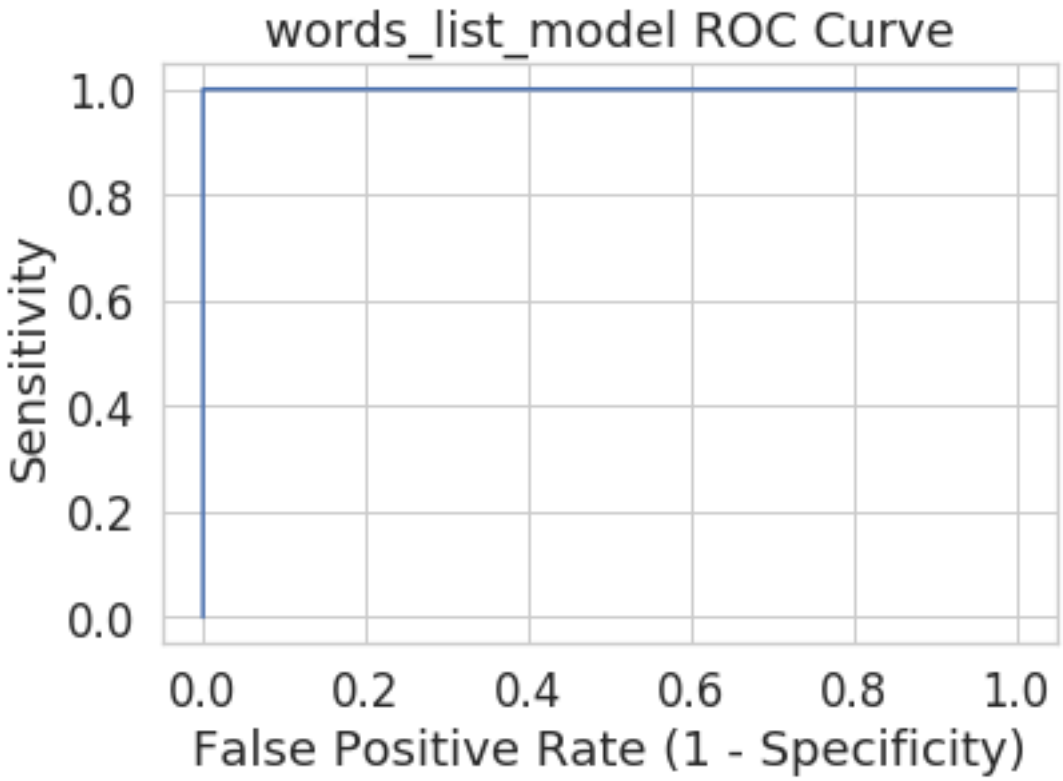
In [174]:
```
from sklearn.metrics import roc_curve

# Note that you'll want to use the .predict_proba(...) method for your classifier
# instead of .predict(...) so you get probabilities, not classes

y_train_predict_proba = model_final.predict_proba(X_train_final)[:, 1]
false_positive_rate_values, sensitivity_values, thresholds = roc_curve(Y_train_final, y_train
plt.step(false_positive_rate_values, sensitivity_values, color='b', where='post')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('Sensitivity')
plt.title('words_list_model ROC Curve')
from sklearn.metrics import roc_auc_score

roc_auc_score(Y_train_final, y_train_predict_proba)
```

Out[174]:
```
1.0
```


words_list_model ROC Curve
Sensitivity
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 (1 - Specificity)