0.0.1 Question 1c

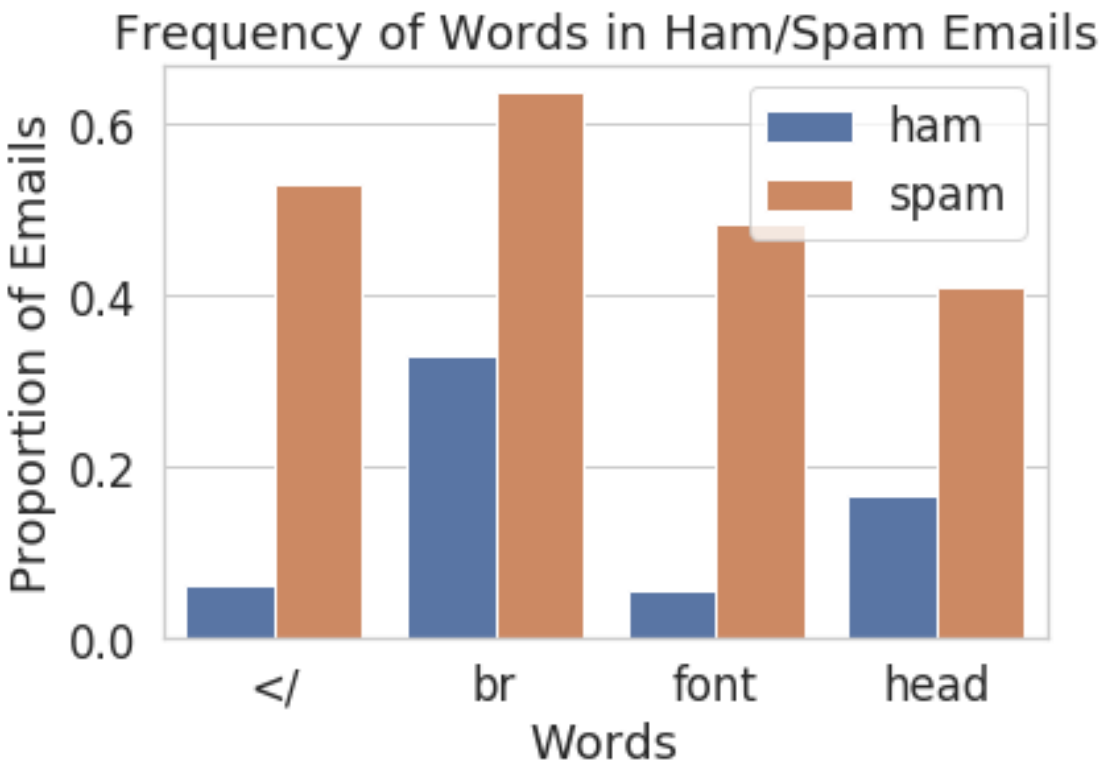
Discuss one thing you notice that is different between the two emails that might relate to the identification of spam.

In the spam email, the html tags, such as <head>, </head>,
 are written in the body of the email. There is also no signature such as "thanks, misha".

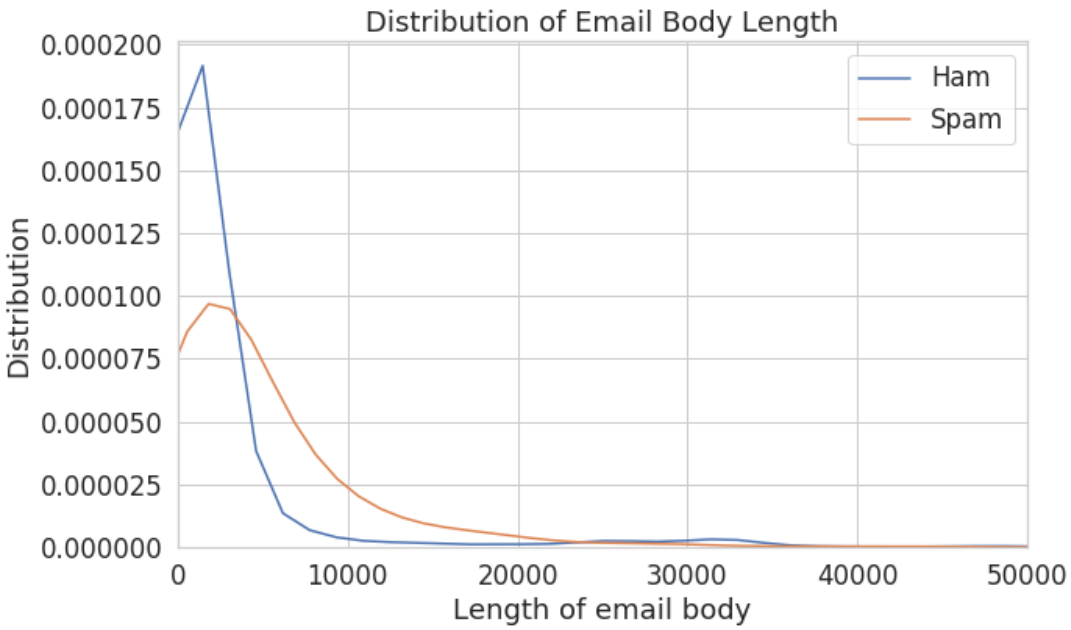
0.0.2 Question 3a

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 [63]: train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of emai
         words = ["</", "head", "br", "font"]
         emails = train["email"]
         spam_words = pd.DataFrame(data=words_in_texts(words, emails), columns=words)
         spam_words["type"] = train["spam"]
         spam_words = spam_words.melt("type")
         spam_chart = spam_words.groupby(["type", "variable"], as_index=False).agg(lambda x: sum(x) / le
         sns.barplot(x="variable", y="value", hue="type", data=spam_chart)
         L = plt.legend()
         L.get_texts()[0].set_text('ham')
         L.get_texts()[1].set_text('spam')
         plt.xlabel("Words")
         plt.ylabel("Proportion of Emails")
         plt.title("Frequency of Words in Ham/Spam Emails");
```



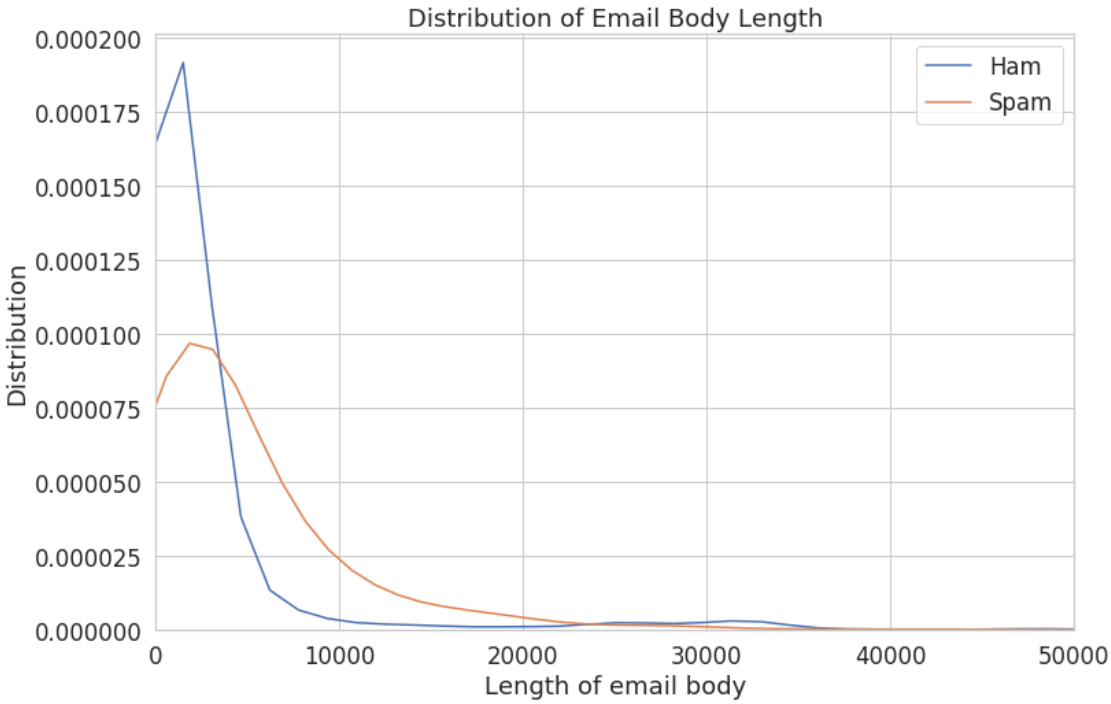
0.0.3 Question 3b



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 [64]:
```
spam_len = train[train["spam"]==1][["email"]]
spam_len["len"] = spam_len["email"].str.len()
ham_len = train[train["spam"]==0][["email"]]
ham_len["len"] = ham_len["email"].str.len()
```

In [65]:
```
plt.figure(figsize=(12,8))
plt.xlim(0, 50000)
sns.distplot(ham_len["len"], hist=False, label="Ham")
sns.distplot(spam_len["len"], hist=False, label="Spam")
plt.legend()
plt.ylabel("Distribution")
plt.xlabel("Length of email body")
plt.title("Distribution of Email Body Length")
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)?

FP is 0 because it measures the number of times the classifier mistakenly predicts positive, when it should have been negative. Since the zero_predictor always predicts 0 (negative) and never predicts 1, FP is 0. FN measures the times an actually positive value was classified as negative. Since everything is predicted as negative in this case, all positive values would have been false negatives. Therefore, it is the number of true positives. Accuracy is the proportion of points the classifier correctly identified. Since the classifier is predicting everything as negative, accuracy in this case would be the proportion of points that are negative in the dataset. Recall is calculated based on the given data, and measures how good the classifier is at detecting positives by penalizing false negatives. Since this classifier never predicts positives, and therefore has 0 true positives, it has 0% recall.

0.0.5 Question 6e

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

The logistic regression classifier has more false positives than the zero_predictor, but has less false negatives. This is because the logistic regression classifier actually predicts some values to be positive, sometimes inaccurately. However, it is also better at detecting positives, so it classifies less observations as negative, resulting in a lower false negative rate.

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. The accuracy for predicting 0 for every email was 74.47%, which is worse than the logistic regression classifier, though not by much. While the zero_predictor had 100% accuracy with classifying negative observations, it failed to identify any positive observations. Considering the accuracy of classifying both positive and negative observations, the logistic regression classifier was overall more accurate.
2. The set of words given are those that could be commonly found in both ham and spam emails. Since there is not a huge distinction between the frequency or proportion of each word found in ham/spam emails, the classifier is performing poorly.
3. I would prefer the logistic regression classifier for a spam filter, because the zero_predictor classifier does not actually detect any spam email and has 0% recall. Because it marks every email as negative (ham), it does not detect positive values (0 TP) and does not filter out spam emails.

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?

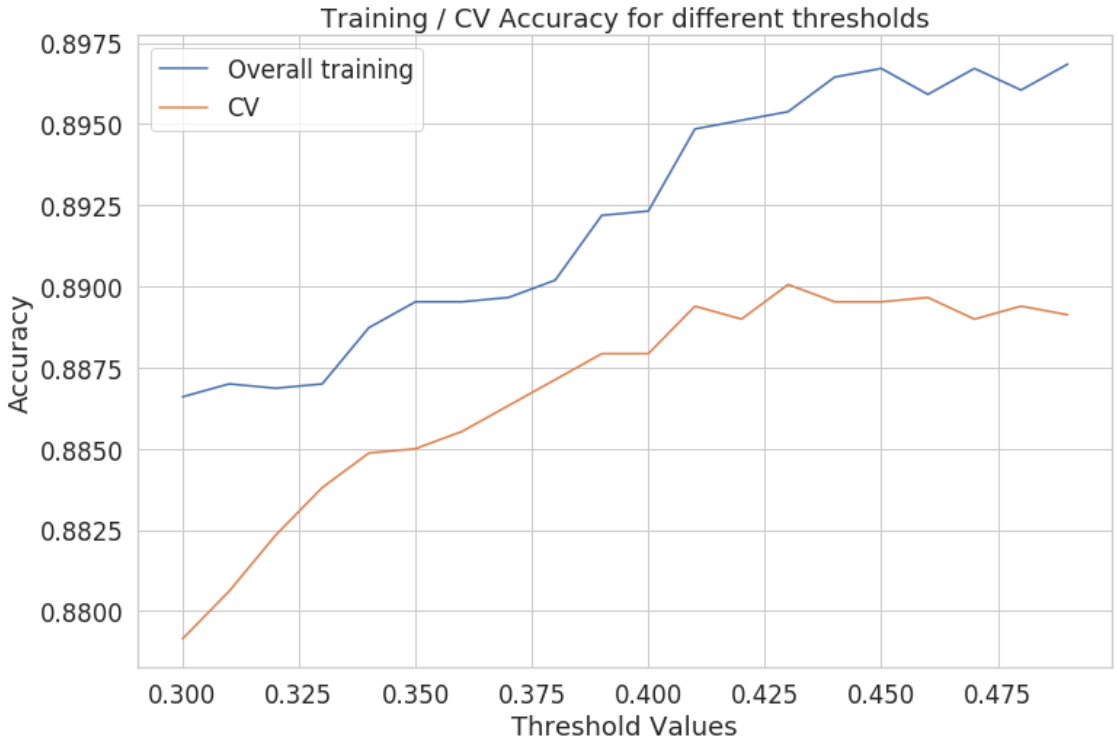
1. I created a dataframe of features and put all the features that I thought would be helpful. Then, I tried adding and/or subtracting features to see if it would change accuracy. I measured the accuracy using the entire training set, and also used cross validation to find the model's accuracy on unknown data.
2. I tried to count the number of various punctuation marks that appeared in the subject line. For example, I tested "! , # $." but "!" was the only punctuation mark that increased the accuracy of the model.
3. While searching for words in the body of the email that would indicate it as spam, I found several spam emails with long chains of #s. I didn't think that the number of "#"s ("#" vs. "###") used to spot such patterns would matter, but using words_in_texts, but "#", "##", "###", "####" all produce different accuracies. After 4 #s, additional #s put in the search string doesn't affect accuracy.

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

In [310]:
```
# Write your description (2-3 sentences) as a comment here:
# Used the ROC curve to approximate a range of optimal thresholds, and plotted an overlapping
# line graph of the overall training set accuracy and the average cross validation accuracies
# to find the threshold that would maximize accuracy. Chose threshold value of 0.43 in the en

# Write the code to generate your visualization here:
plt.figure(figsize=(12,8))
plt.plot(threshold_list, test_accuracies, label="Overall training")
plt.plot(threshold_list, CV_accuracies, label="CV")

plt.xlabel("Threshold Values")
plt.ylabel("Accuracy")
plt.title("Training / CV Accuracy for different thresholds")
plt.legend();
```



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.

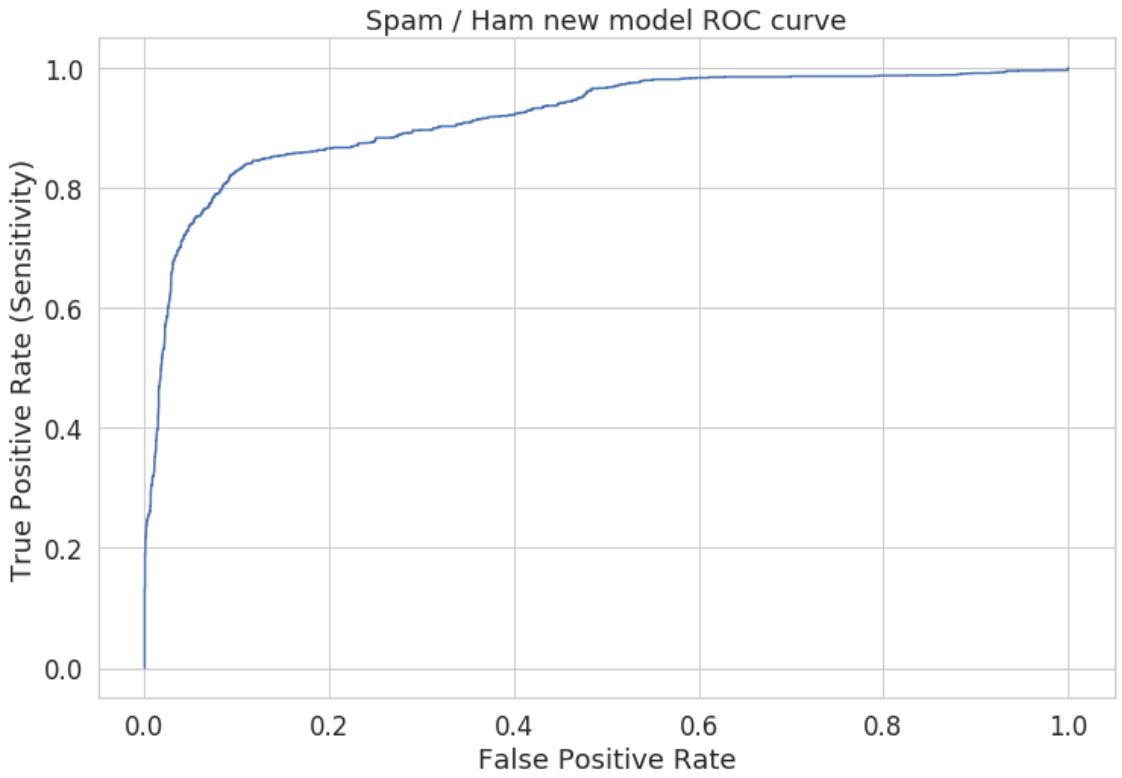
Recall that logistic regression calculates the probability that an example belongs to a certain class. Then, to classify an example we say that an email is spam if our classifier gives it ≥ 0.5 probability of being spam. However, *we can adjust that cutoff*: we can say that an email is spam only if our 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 [311]: from sklearn.metrics import roc_curve
          import plotly.express as px


          # 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, threshold = roc_curve(train_Y,
                                          new_model.predict_proba(train_X)[:, 1])
          plt.figure(figsize=(12,8))
          plt.plot(fpr, tpr)
          plt.xlabel("False Positive Rate")
          plt.ylabel("True Positive Rate (Sensitivity)")
          plt.title("Spam / Ham new model ROC curve");
```



In [312]:
```
fig = px.line(x=fpr, y = tpr, hover_name=threshold)
fig.update_xaxes(title="False Positive Rate")
fig.update_yaxes(title="True Positive Rate")
fig
```