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.

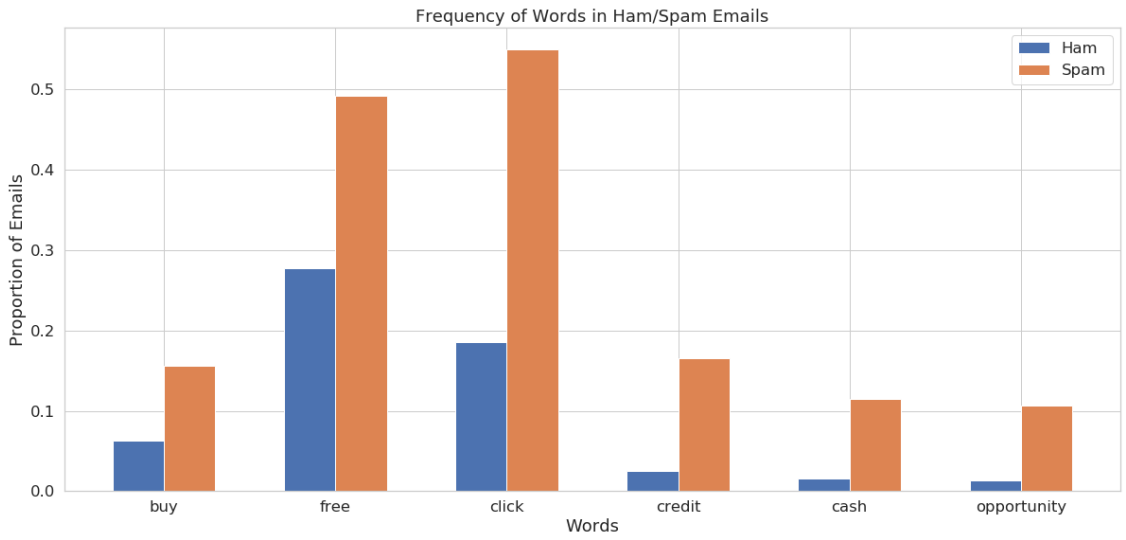
The spam email is written in html while the ham email is text in a string.

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 [304]:
```
train=train.reset_index(drop=True) # We must do this in order to preserve the ordering of ema

spam_words = ['buy', 'free', 'click', 'credit', 'cash', 'opportunity']

num_spam = train['spam'].sum()
num_ham = len(train['spam']) - num_spam

ham_emails = train[train['spam'] == 0]['email']
spam_emails = train[train['spam'] == 1]['email']
ham_ind = words_in_texts(spam_words, ham_emails)
spam_ind = words_in_texts(spam_words, spam_emails)

plt.figure(figsize=[20, 9])
plt.bar(x=spam_words, align='edge',height = np.sum(ham_ind, axis = 0) / len(ham_emails), label
plt.bar(x=spam_words, align='edge',height = np.sum(spam_ind, axis = 0) / len(spam_emails), la
plt.legend()
plt.title('Frequency of Words in Ham/Spam Emails')
plt.xlabel('Words')
plt.ylabel('Proportion of Emails')
```

Out[304]:
```
Text(0, 0.5, 'Proportion of Emails')
```

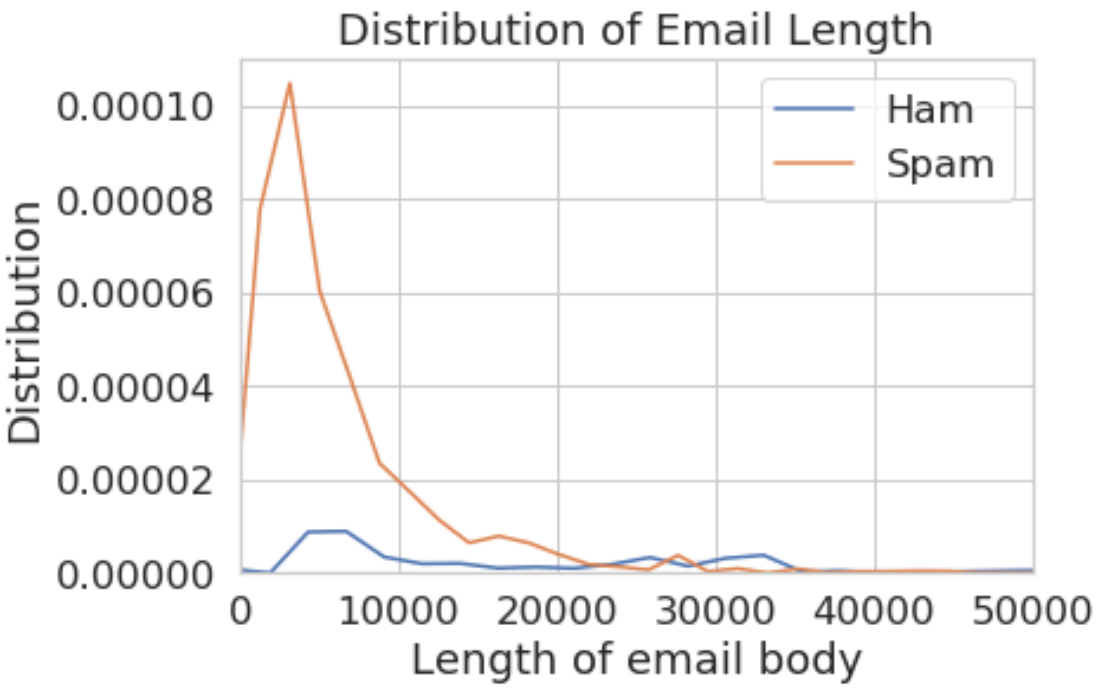


0.0.3 Question 3b

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 [305]: ham_length = [len(email) for email in train[train['spam'] == 0]['email']]
          spam_length = [len(email) for email in train[train['spam'] == 1]['email']]

          plt.xlim(0, 50000)
          sns.distplot(ham_length, label = 'Ham', hist=False)
          sns.distplot(spam_length, label = 'Spam', hist=False)
          plt.legend()
          plt.xlabel("Length of email body")
          plt.ylabel("Distribution")
          plt.title("Distribution of Email Length")
```

```
Out[305]: Text(0.5, 1.0, 'Distribution of Email Length')
```



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)?

For 6a, since the zero predictor always predicts negatives (ham), there are 0 false positives, and the false negatives are the number of emails that are actually labelled spam in the dataset. For 6b, the accuracy is just the proportion of emails labelled ham in the dataset, since our predictor will always predict ham emails correctly. The recall is 0 beacuse our model never predicts any positives so the number of true positives is 0, which makes recall 0.

0.0.5 Question 6e

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

Since recall is much smaller than precision, there are more false negatives. Also, simply counting false positives and false negatives, we see fp = 122 and fn = 1699.

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 zero predictor only has an accuracy of 74.5% so the logistic regression classifier is better.
2. Some of the words given like 'memo' may not appear much in both ham and spam emails, while words like 'private' may appear in roughly equal proportions in both ham and spam emails.
3. I would prefer using the logitic regression model for the spam filter because it has the higher prediction accuracy (75.6%). Additionally, the 0 predictor predicts all emails as ham and has a recall of 0 meaning that it will let all spam emails into an inbox, whereas the logistic regression model will filter out some of the spam.

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:

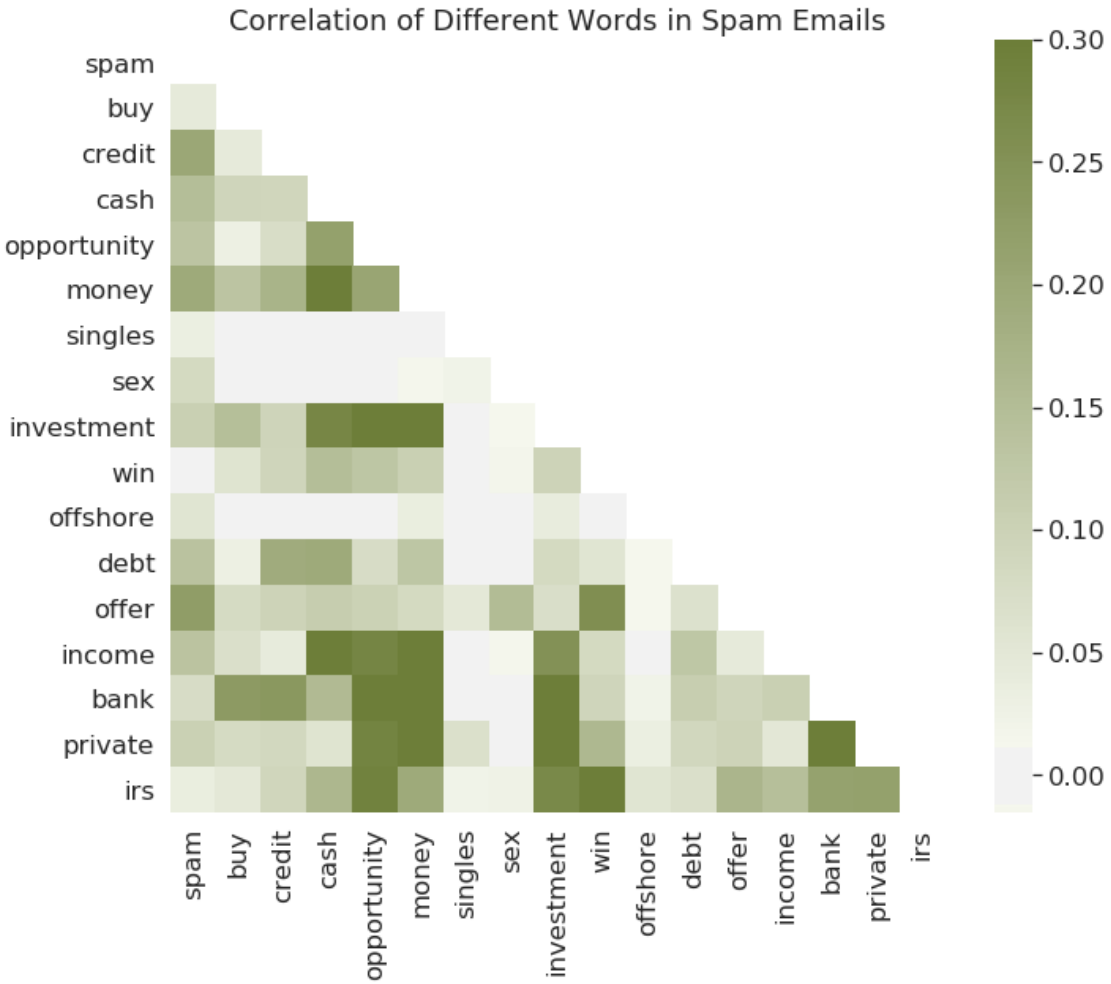
1. How did you find better features for your model?
2. What did you try that worked / didn't work?
3. What was surprising in your search for good features?

1. I used several of the suggestions listed above (number of characters in email, if it was a reply, forwards, etc.) and looked and bar plots and overlapping density curves to see the differences in the distribution of ham and spam emails with a specific feature. I also combine words from problem 3 which I believed were common in spam email (domain knowledge) as well as looking at the most common words present in spam and ham emails in the data set.
2. I tried looking at the proportion of capital letters in the email (not the subject) but that was too low as both ham and spam emails dont have too many capital letters within the email body itself. Also, when I used the number of letters in the body of an email, it reduced both training and validation accuracy so it was ultimately removed from the features that I used. This was probably due to the fact that the distributions of number of characters in the email body for spam and ham emails were to similar, or perhaps because ham emails tend to very in length quite a bit.
3. I was surprised that words indicating visual media (like src, img, and gif) were highly common in ham emails and not nearly as common in spam emails as I would think that these type of things would be involved in advertising. I also was surprsied at how much more the ! character appeared in spam emails than in ham email, because I thought the distributions of ! in ham and spam would be similar as ! could be useful for getting attention (for spam) but also in a congratulatory way (used in ham), but this was not the case.

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

In [326]:
```
# Write your description (2-3 sentences) as a comment here:
# As some of the features, I included the count of a few words that are commomly associated w
# The plot below shows the correlations between the occurances of these words within the set
# Darker squares represents words with higher correlations, inciating that the words co-occur
# For instance, the square at the intersection of 'bank' and 'investment' is relatively dark,
# of words appears somewhat frequently in the same piece of spam. For this reason, they may c
# cause multicolinearity which can lead to the feature weight varying wildly. This can most l
# designing a feature that caputures both the idea of 'bank' and 'money' such as the word 'lo
#
#

# Write the code to generate your visualization here:

plot_data = pd.DataFrame()
plot_data['spam'] = train['spam']
meaningful_spam_words = ['buy', 'credit', 'cash', 'opportunity',
                         'money', 'singles', 'sex', 'investment',
                         'win', 'offshore', 'credit',
                        'debt', 'offer', 'income', 'bank', 'private', 'irs']
for w in meaningful_spam_words:
    plot_data[w] = train['email'].str.count(w)
corr = plot_data.corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(100, 100, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0)
plt.title('Correlation of Different Words in Spam Emails')
```

Out[326]:
```
Text(0.5, 1, 'Correlation of Different Words in Spam Emails')
```


Correlation of Different Words in Spam Emails
spam
buy
credit
cash
opportunity
money
singles
sex
investment
win
offshore
debt
offer
income
bank
private
irs
0.30
0.25
0.20
0.15
0.10
0.05
0.00

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 [327]:
```
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

model = LogisticRegression(max_iter=1000)
X_train_plot = design_matrix(train)
Y_train_plot = train['spam']
model.fit(X_train_plot, Y_train_plot)

cutoff = np.linspace(0, 1, 100)

true_pos_rate = []
false_pos_rate = []

y_hat = model.predict_proba(X_train_plot)
for c in cutoff:
    y_hat_2 = y_hat[:, :1].reshape(1, len(y_hat))[0]
    y_hat_2 = np.where(y_hat_2 < c, 1, 0)
    y_obs = Y_train_plot.values

    tp = sum((y_obs == y_hat_2) & (y_obs == 1))
    fn = sum((y_obs != y_hat_2) & (y_obs == 1))
    tpr = tp / (tp + fn)

    fp = sum((y_obs != y_hat_2) & (y_obs == 0))
    tn = sum((y_obs == y_hat_2) & (y_obs == 0))
    fpr = fp / (fp + tn)

    true_pos_rate.append(tpr)
    false_pos_rate.append(fpr)

plt.plot(false_pos_rate, true_pos_rate)
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title("ROC Curve")
```

```
Out[327]: Text(0.5, 1.0, 'ROC Curve')
```