Alex Beeston and Colby Wight
Data Science Incubator CS 5830/6830

Project5 - Naive Bayes

Introduction

Keeping emails organized can be a tough job. With likely hundreds of emails coming in over the course of a week, going through emails individually can be quite time consuming. We set out to develop an email filtering system to see how well we could detect promotional emails. Having these in a separate folder from other emails can help reduce clutter and save time. We used our own emails to train and test a naive bayes classifier. We found that we could differentiate emails fairly well using a combination of features, but work is still needed to put this into production.
Project slides:
https://docs.google.com/presentation/d/13B1PvhGJYJ3orP-Xu1g21brzUXuZJbadg88naqg6ZGk/edit#slide=id.g7eba8516b1_0_21

Dataset

Originally, we were going to combine our promotional and regular emails from both of us into a single dataset. However, we found that it was difficult to combine the email files into a single dataset, presumably because we tried to send the emails over an email as attachments. To compensate for this, we used our analysis technique on each of our data sets individually. Therefore, the results section is divided into the results for Alex's emails and the results for Colby's emails. The models for each dataset will not necessarily generalize well to other sets of emails, but they perform well against our own data.

The data sets were generated by downloading forty emails in .eml format from the "promotions" and "inbox" mailboxes of our gmail accounts. The python module "email" was then used to parse the .eml files and build a tree structure from which we could generate a feature set.

Analysis Technique A Multinomial Naive Bayes Classifier was used to classify the email as regular and promotional email. We went with multinomial over gaussian because all of our features were categorical. A variety of features were used to determine which feature sets performed best. To compute F1 scores, precision, and recall, we randomly partitioned our data into training and testing sets 100 times and took averages for each of these metrics across the 100 trials.

Using Colby's emails we conducted a systematic study of all of the proposed attributes. We created a classifier using only one attribute at a time and calculated the performance metrics as before. Once we got an idea of what features worked well for this set of emails, we tried a few different attribute sets.

Results from Alex's emails

For Alex's emails, the following feature sets were tested:

- Feature Set A: contains the word "sale", contains the word "now", contains the word "free", and the day of the week when the email was sent
- Feature Set B: contains the word "buy", contains the word "now", and the month of the year when the email was sent

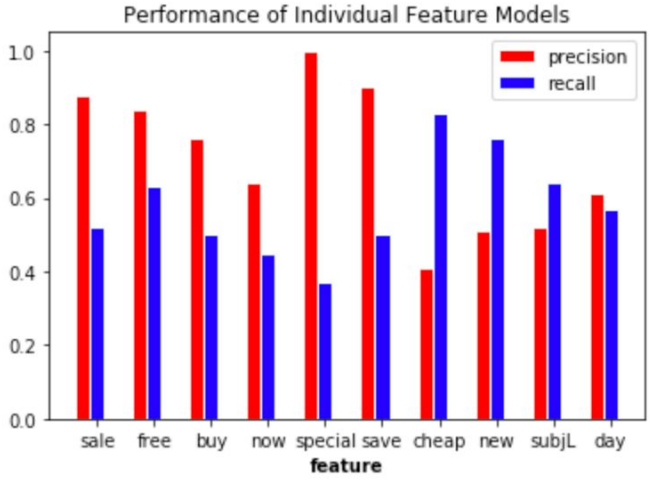
	Feature Set A	Feature Set B
Promotion F1 Score	0.636	0.825
Promotion Recall	0.615	0.826
Promotion Precision	0.685	0.843
Inbox F1 Score	0.684	0.819
Inbox Recall	0.727	0.834
Inbox Precision	0.662	0.830

Although Feature set B appears to be a better predictor for promotional email, it should be noted that the dataset is comprised of the forty most recent promotional and regular emails only, which sum to a mere eighty emails. Consequently, emails from only December, January, and February were in the dataset. Although classifying on this feature could be promising, a larger data set would be required to generate conclusions about the viability of predicting promotional email according to the month of the year in which it was sent. However, it can be concluded that this classifier predicts promotional email from regular email moderately well against its own data.

Results from Colby's emails

When each variable was tested individually the word special obtained the best promotion precision at 1. This means that everytime the model predicted an email as being a promotional email it was right. On the other side, the promotional recall score was .37 meaning we only found 37% of the actual promotional emails. Why did this happen? Our guess as to why this happened is that none of the normal emails contained the word special. Since we only had 40 emails in the dataset this is very possible. Many other single variable models followed a similar trend. Using the presence of a single word as the attribute we were able to get a good promotional precision (as well as a good inbox recall) but we were never able to get promotional

recall higher than 63%. See the grouped bar graph below for precision and recall using different variables.



This got us thinking. What would happen if we created a model with some of the features that had very high individual precision? Would the precision stay high? Would the recall get brought up as well? For the first model we added all the attributes that had a precision greater than .5, for the second model we added all the attributes that had a precision greater than .8. The baseline model had all of our attributes.

We found that the best model was the one that used the attributes with precision greater than .5 (‘sale’, ‘free’, ‘buy’, ‘special’, ‘save’). All models had the same or worse recall and this one had the best precision. Indeed we see that the recall was brought up quite substantially while the loss in precision was not terrible. It is still unfortunate that some of the actual emails are getting labeled as promotional. This is not good for some who may miss these emails and have to search through the promotional box. Therefore we are not quite ready to use this on our own emails yet. Next we need to find more attributes that will help in keeping normal emails from going to the wrong inbox.