Assignment 2 - Emails For SPAM/HAM Classification

Yifu Wu (916619585) *

11/01/2020

I completed this assignment myself without consulting books or tutorials that specifically addressed this data set.

In recent time, using statistics method to classify an email as SPAM or HAM has become a hot topic. However, it is hard to directly identify whether it is a spam email. So we first need to process each mail message to a list which contains the header, the body and the attachments (if any), then we should derive variables which give us measures of each email. By using plots and building classification models, we can successfully discriminate between SPAM and HAM email messages. In this report, I will introduce the approach that I use to identify SPAM email messages.

1 Processing and Listing Email Messages

There are total 6541 files, and we should first figure out whether all these files are email messages. Since the header of the email is in a special "KEY : VALUE" format, we could find out the pattern using the regular expression "ˆ[A-Za-z][-A-Za-z]+:" and grepl function. We find that all files start with "From xxx" or "KEY : VALUE" pattern except one, which is obviously not an email, so we remove it from the list. In this section, I will introduce the approach to split an email message into three parts, which are header, body, and attachments. Although I cannot ensure the approach I use can process all email messages without any error, I have tried my best to consider as much as possible.

1.1 Process Header

Since we only need "KEY : VALUE" format header to derive variables in the latter part, we first need to remove those lines start with "From xxx" using grepl function with regular expression "ˆFrom .* [0-9]{4}$". One tricky part is some "From xxx" do not appear in the first line of the email, so we use grep function to locate the pattern. Then the header will be the first line until the first blank line. However, it is more convenient to store the header as the data frame, which could use read.dcf together with textConnection function. Then we can regard each as an observation with variable name (KEY) and corresponding values.

1.2 Process Boundary

The boundary is a crucial part to connect the header and attachment, which means we cannot locate and separate attachments without the boundary in the email. Since email messages are different from each other, we need to use various ways to grab different type of boundaries.

The first type is more straightforward, we can use the "Content-Type" or "Content-type" field in the header since it contains the boundary if the email message has one or more attachments. Most

*Department of Statistics, UC Davis and ifuwu@ucdavis.edu

of the email has the format: boundary="xxx", so we split the string by "boundary=", and the second separated part will be the boundary. We cannot get the boundary after splitting the string since R will also add slash before the symbol like double quotation marks, to get the real boundary, we need to use gsub function to remove the slash and double quotation marks. However, some fields can have the format like: boundary="xxx";charset="xxx", in this case, based on the steps above, we can remove the part after the semicolon by using gsub function with regular expression: ";.*". There is also a format: boundary=xxx. In this uncommon case, the double quotation marks disappear, and we can easily locate the boundary by selecting the second part after splitting the string by "=".

The second type is tricky, we cannot find the content type in the header, but it does have one or more attachments. In this case, we need to fulfill two conditions in order to find the boundary. First, we should find the "MIME-Version" fields in the header, then we should grab the pattern with regular expression: "^[-]2.+[a-zA-Z0-9]+$" since the indicator of the start of an attachment is "- -"+boundary. So, we just need to find the line with the the pattern in the part after the header and then remove the first two dashes ("- -").

1.3 Process Body

The next part is to process the body of the email. The body of an email is all text after the first blank line following the header and up to any attachments, also there may be some body text appear behind the attachments. It is a little bit complicated since there are several different situation in our data.

The first situation is straightforward, if the output of the processing boundary section indicates that there is no boundary, we can simply consider that the email does not have any attachment, and everything exclude the header is the body.

The second situation is that we find a boundary in the previous section which means that there exists at least one attachment. We know that the boundary string marks the beginning of each attachment, and it is prefaced with two additional hyphen. Also this type of string is used as the end of the attachment part, but it has two additional hyphens at the end of the string. We can use both the start and end string to locate attachments in the text. In this situation, we first consider the case with the ending string. If there are both starting and ending string in the attachment, we could easily identify the attachment, and we should grab the pattern with grep function. Since there may exist some symbol issue, we should set the argument "fixed=TRUE". However, there exists another situation that the boundary in the header could not perfectly match the boundary in the text. For example, we have "boundary=xxx" in the header, while we have "boundary= xxx", which has a addition space after the equal sign. In this case, we should use agrep, which is the approximate grep function to match the string with the same regular expression. After confirming the starting and ending point of the attachment, the texts exclude those parts are all body.

The third situation is rather similar with the second one, however, there is no ending string in the attachment. In this case, we regards the last line of the email as the end of the attachment, and the starting line can be identified using the same way in the previous situation. The only different is the ending indicator of the attachment. Then we can simply get all body text by removing all the attachment part.

1.4 Process Attachments

The next part is to process the attachment of the email, in the previous part, we have successfully identified the attachment. In this part we will separate attachments if there is more than one attachment in the email. The attachment also have the header and body, we can use the same way

before to divide them by using the first blank line in the attachment. However, some attachments to do have a clear boundary of header and body, even few of them does not have any header. Since we do not need to divide the body and header in the latter part, so we just keep two parts (header and body) together, and consider it as a complete attachment.

The first situation is straightforward, if the output of the processing boundary section indicates that there is no boundary, we can simply consider that the email does not have any attachment. In this case, we just output a NULL list.

Similar with the processing body part, the second situation is that both starting and ending boundary exist. After getting the starting and ending boundary using the approach mentioned in the previous section, the only thing we should do is to list all attachments (maybe more than one attachment exist in an email). If there is only one attachment, we can say that all text between starting and ending boundary is the attachment. If there are two or more attachments, the attachment will be the text between two starting boundary. We can simply use a for loop to fulfill it.

The third situation is that there is no ending boundary. The method to extract the attachment is the same as the previous one. The only thing we should aware is that the last line of the email is the ending boundary of the last attachment.

Up to now, we have successfully got the header, the body and the attachment for each email. What we should do is to use lapply function to apply the approach on each email. Here we get a large list contains 6540 elements, and for each element it is a list contains three element. In the next section, we can use this large list to derive variables.

2 Derive Variables from Email

In this section, 22 variables which are derived from the email messages will be introduced. 12 of them are logical variable, 9 of them are numerical variable, and the last one is the categorical variable.

2.1 isSpam

The isSpam is a logical variable which indicates the mail is spam or ham. This one is pretty simple, we can just grab "spam" from the folder name to decide whether it is a spam or not.

2.2 isRe

The isRe is a logical variable which indicates that whether string "Re:" appears as the first word in the subject of the message. In this case, we use grepl function with regular expression: "Re:" to find whether the subject of an email contains "Re:". Also we set the argument ignore.case=TRUE, which we can ignore the upper or lower case. The reason is that since some subject does not start with "Re:" but the email is to reply another one, and it does contains the string like "RE:" or "re:". Thus, we also regard those as an indicator of replying an email.

2.3 numLinesInBody

The numLinesInBody is a numerical variable which count the number of lines in the body of an email message. For this one, we can first take the body part out from the large list and then count the number of lines by using length function.

2.4 bodyCharacterCount

The bodyCharacterCount is a numerical variable which count the number of characters in the body of an email message. Since the body of an email has several lines, and we can simply use nchar function to calculate each line and sum them up.

2.5 isYelling

The isYelling is a logical variable which indicates that whether the subject of the mail is in capital letters. We can use grepl function with regular expression "[a-z]" to check whether there exists lower case letter in the subject, and we can return TRUE if no lower case letter is found in the string.

2.6 numDollarSigns

The numDollarSigns is a numerical variable which indicates the number of dollar signs in the body of the message. In order to count the number of dollar signs much easier, we just use paste function to combine all lines into a single string. Then we use gregexpr function with regular expression "\\$" to find and count the number of dollar signs in the body.

2.7 numAttachments

The numAttachments is a numerical variable which indicates the number of attachments in the message. Since we store the attachments as list, so it is easy to count the number of attachments by calculate the length of the list. If the list is NULL, the number of the attachment will be 0.

2.8 subjectExclamationCount

The subjectExclamationCount is a numerical variable which indicates the number of exclamation marks in the subject of the message. Similar with the dollar signs, we use gregexpr function with the regular expression "\\!" to find and count the number of exclamation in the subject.

2.9 isDear

The inDear is a logical variable which indicates whether the message body contains a form of the introduction Dear. By my understanding, the introduction dear will appear at the first place of a line. Thus we do not count any dear appears in the middle of a line. So we use the regular expression "^Dear" with grep function to identify whether any line of the body has an introduction dear.

2.10 subjectSpamWords

The subjectSpamWords is a logical variable which indicates whether the subject contains one of the spam phrase. We can use grepl function to discover whether those phrases appear in the subject. The phases include: viagra, pounds, free, weight, guarantee, millions, dollars, credit, risk, prescription, generic, drug, money back, credit card.

2.11 subjectQuestCount

The subjectQuestCount is a numerical variable which indicates the number of question marks in the subject. We can use the same approach as the subjectExclamationCount to count the number. The regular expression in this case is "\\?".

2.12 multipartText

The multipartText is a logical variable which indicates whether the header states that the message is a multipart text. This one is a little bit complicated than others. Since the "multipart" always appears in the content type field in the header, so if the header does not contain the content type, we can say that it not states that there are multi-attachment in the email. If we find the multipart character in the content type by using the grepl function with regular expression "multipart\", we can say that it states that email has more than one attachments. However, some email do not have multipart character in the content type field but it does have multiple attachments, we also regard those cases as TRUE in this variable. Although it does not directly indicate there are several attachments in the text, the boundary does states there exists attachment (one or more). So we also consider those cases as TRUE.

2.13 isInReplyTo

The isInReplyTo is a logical variable which indicates whether the header of the message has an In-Reply-To field. This one is pretty simple, we can check whether "In-Reply-To" is in the column name of the header by using "%in%".

2.14 priority

The priority is a categorical variable which indicates the level of the priority of an email message. We find that the "X-Priority" (no "X-Smell-Priority" field in the header) field in the header has three levels. Number 3 indicates that the priority level is normal and number 2 indicates that the level is high. However, part of the number 1 indicates that the level is high, while the other part indicates that the level is the highest. So, I just regard number 3 as normal and both number 1 and number 2 as high. This setting is reasonable since both number 1 and 2 can represent the high level of priority, and it is not appropriate to set 2 as middle just because we cannot downgrade the level. To implement the approach is easy, the only thing we need to do is to grab the corresponding number and assign a priority level to that email.

2.15 numRecipients

The numRecipients is a numerical variable which indicates the number of recipients in the "To", "Cc" fields. The first step is to know how many situations will appear in this case. We can have both "To" and "Cc" fields, or have one of the two fields in the header, and we can also have neither fields. We use "@" as the indicator of a recipient, and we want to calculate how many "@" appears in two fields. So we use gregexpr function with the regular expression "\\@" to count the number of recipients. Although it may not be the most robust approach to count the number of recipient, it is a easiest way with acceptable error.

2.16 isOriginalMessage

The isOriginalMessage is a logical variable which indicates whether the body contains the phrase "original message" or something similar. In this case, I search two different character which are "original message text" and "original message" since they always appear when the message is original. We use grepl function with argument ignore.case=TRUE since we do not know which letter in the string is capitalized or not.

2.17 fromNumericEnd

The fromNumericEnd is a logical variable which indicates whether the user login in the "From:" field ends in numbers. In this case we need to find out whether the last character in the "From" field before the "@" character is number or not. We can use the regular expression "[0-9]+@" with the grepl function to check it out.

2.18 isPGPsigned

The isPGPsigned is a logical variable which indicates whether the mail was digitally signed. Since "PGP" and "GPG" are very uncommon words in an email, we can simply search the "PGP" and "GPG" in the body and the attachment of the message.

2.19 percentSubjectBlanks

The percentSubjectBlanks is a numerical variable which indicates the percentage of blanks in the subject. In this case we can calculate the percentage by using "number of the character of blank in the subject / total number of the character in the subject". We first use nchar to calculate the total number of the character (it also include the blank), and then we use the gregexpr function the calculate the number of blank in the subject.

2.20 subjectPunctuationCheck

The subjectPunctuationCheck is a logical variable which indicates whether the subject has punctuation or digits surrounded by character. In this case, we need to have two regular expression which are "[a-zA-Z]+[[:punct:]]+[a-zA-Z]+" and "[a-zA-Z]+[0-9]+[a-zA-Z]+". Then we use grepl function to find out whether these two patterns appear in the subject.

2.21 replyUnderline

The replyUnderline is a logical variable which indicates whether the Reply-To field in the header has an underline. In this case we need to find the underscore "_" in the "Reply-To" field. So we just need to use grepl function with regular expression "_".

2.22 hourSent

The hourSent is a numerical variable which indicates the hour in the day the mail was sent (0-23). This one is a little bit tricky. Since the format of the date is not identical, so we use str_extract function in the stringr package to extract the time in the "Date" field with the regular expression "[0-9]+:+". Then we use strsplit function with argument split=":", the first string we get is the hour of the day. Although the timezone of each email may not be the same, we can analysis the time of sending the email. For example, we can say that most of the spam email is sent at 7 pm local time.

3 Explore the Data

In this section I will explore the variable which has been derived in the previous section. Since some variables cannot help to discriminate between spam and ham email messages, we just pick some interesting and useful variables that can be used to separate different type of email messages. The plots and numerical summaries of other variables can be found in the R script.

3.1 Categorical Variable and Logical Variable

In this section, I will introduce some categorical and logical variables that can be used to discriminate between spam and ham emails. Other variables may not have much difference between these two type of email. In this dataset, we have 4864 ham emails, and 1676 spam emails, since the number of two kinds of email are not the same, we use relative frequency to analysis the relationship between isSpam (Whether the email is spam) and other variables.

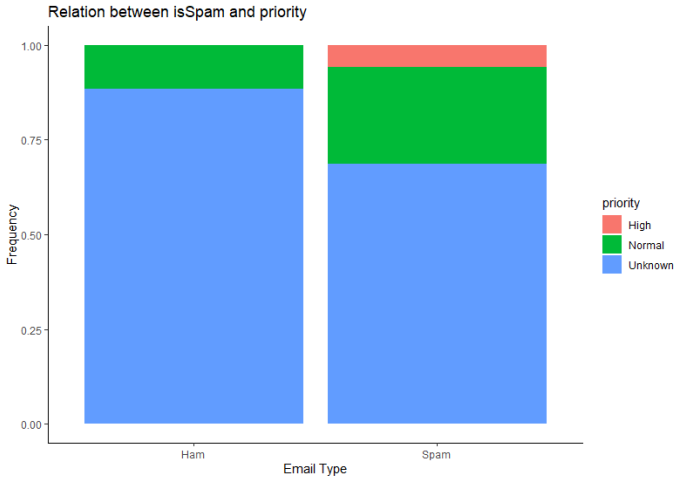

Figure 1: Relation between isSpam and priority

From the figure 1 above we know that priority level of an email can be an indicator of spam email. Since we know that for the ham email messages, there is almost no high level of priority in ham emails, while for the spam email, about 6% of them need to be sent in high priority. We know that people who send spam email want more recipients to read the email as soon as possible, which lead to the high priority. We can say that if an email is sent in high priority, it is more likely to be a spam.

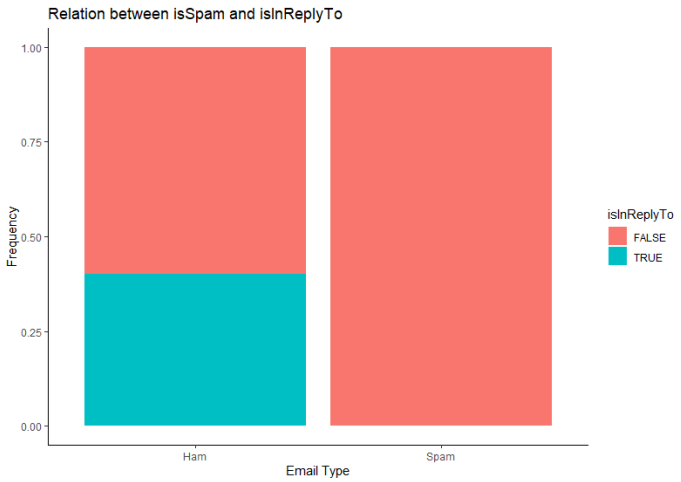

Figure 2: Relation between isSpam and isInReplyTo

From the figure 2 above we know that whether the header of an email has the "In-Reply-To" field can be an indicator to distinguish between spam and ham. We know that about 40% of the

ham email do have the "In-Reply-To" field, while no spam email have that field. "In-Reply-To" field indicates that the email is a reply to the previous email. Since no one wants to reply a spam email, we can definitely say that if an email has an "In-Reply-To" field, it is a ham email.

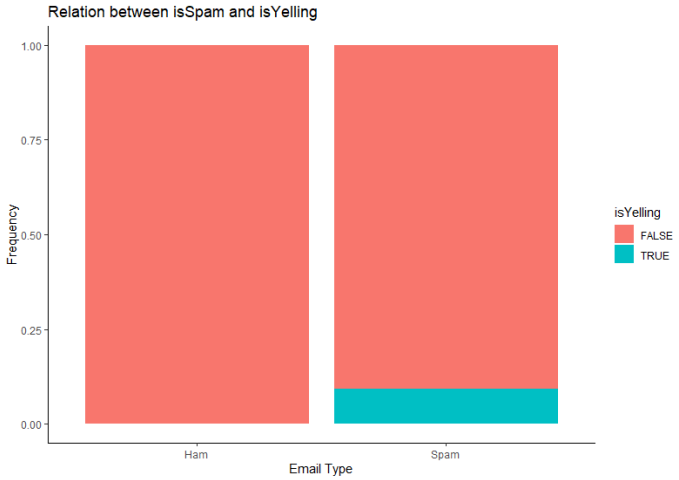

Figure 3: Relation between isSpam and isYelling

From the figure 3 above we know that for the subject of spam email, about 10% of them are in capital letters, while for ham email, only 0.1% are in capital letters, which can be ignored. However, most of the spam email still have lower case letters in the subject, so we just can say that if all letters in the subject is upper case, it is more likely to be a spam email. If the subject has lower case letters, we can not directly judge whether it is a spam email or not, more information is needed. Same situation happens in the other variables, such as "isDear", "subjectSpamWords", "fromNumericEnd" and "replyUnderline".

3.2 Numerical Variable

In this section, I will introduce some numerical variables that can be used to discriminate between spam and ham emails. I will use histogram to explore the data since it can give me the number of elements in a certain range.

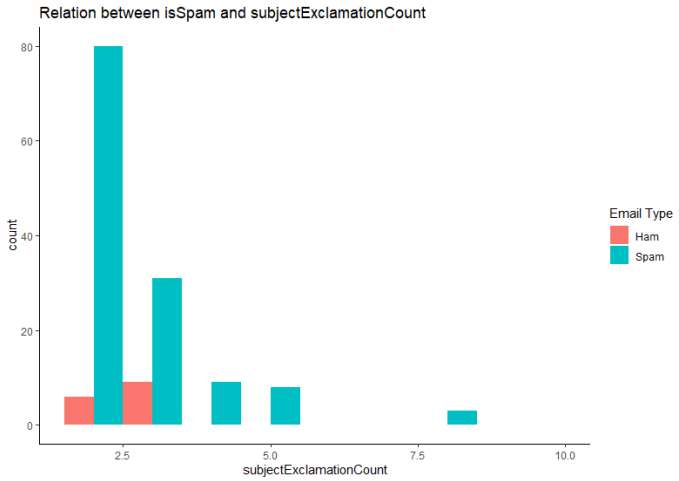

Figure 4: Relation between isSpam and subjectExclamationCount

The figure 4 shows the relationship between the "isSpam" and "subjectExclamationCount" (number of the exclamation in the subject). Here, we just care about the elements which have two or more exclamation in the subject, since in most subjects there only exists one or less exclamation. From the plot we can see clearly that in this case most of the messages are spam. So we can say that if there are two or more exclamation in the subject, it has a great possibility to be a spam email.

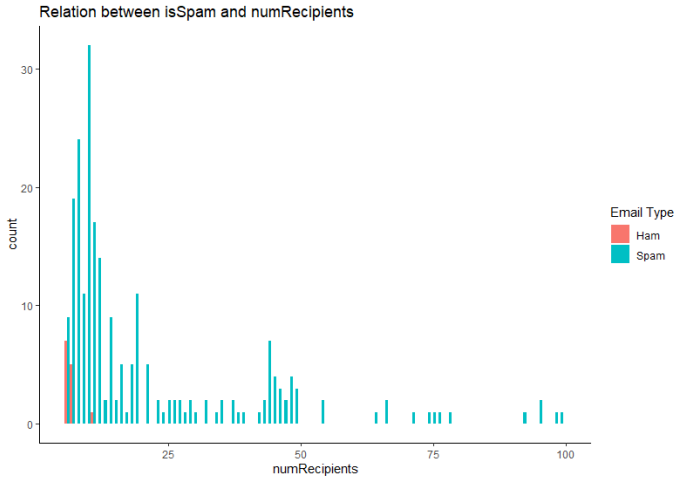

Figure 5: Relation between isSpam and numRecipients

The figure 5 shows the relationship between the "isSpam" and the number of recipients of an email. Here we care about the email message which has 6 or more recipients, since a regular email often has few recipients (less than 6). From the plot we know that spam email always has a large number of recipients since the sender wants more people to receive the email (maybe introduce the product, attract people to apply for a credit card, etc.). So we can say if the number of recipients of an email is large, it is more likely to be a spam email. However, we cannot give the a very precise boundary to discriminate spam and ham email since it is possible for a ham email to have a huge number of recipients, for example, the email send by school or company.

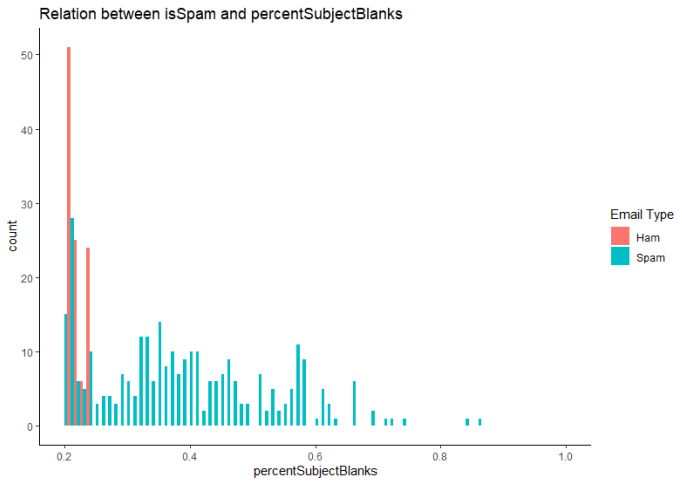

Figure 6: Relation between isSpam and percentSubjectBlanks

The figure 6 above shows the relationship between "isSpam" and the percentage of blank in the subject of an email. From the plot we know that the percentage of the blank in the subject of most ham email messages are lower than 20%, and we can observe that if more than 25% of the character in the subject is blank, it is a spam email. So the more the percentage of blank in the subject, the email is more likely to be a spam.

Other numerical variables like "numLinesInBody", "bodyCharacterCount", "numDollarSigns", etc. cannot be used as the indicator to identify spam or ham, since the distribution of the value for two different type of email are similar.