

Natural Language Processing

MANDATORY PROBLEM (COURSEWORK 7.3)

JIMMY TAN

00819296

Introduction

I have implemented a spam filter based on naive Bayes in this project. The data set used for the training and testing of the algorithm can be found in the UCI Machine Learning repository using the following link:

<https://archive.ics.uci.edu/ml/datasets/sms+spam+collection>

This data set contains 425 SMS spam messages that were manually extracted from the Grumbletext website and 3375 randomly chosen ham messages from the NUS SMS Corpus (NSC) from the National University of Singapore. Therefore, in this data set, you can see a lot of informal English that is widely used by Singaporeans and Malaysians. For example:

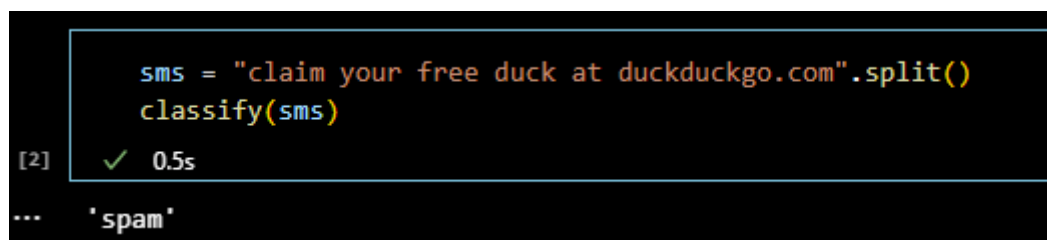
- Ok lar... Joking wif u oni...
(Don't be mad, I was just joking with you only)
- U dun say so early hor... U c already then say
(Don't make conclusions so early before you see it)

The rest of the data set consists of 450 ham messages from Caroline Tag's Ph.D. Thesis, 1002 ham messages and 322 spam messages from SMS Spam Corpus.

Goal of the project

The goal of the project is to classify whether a message (it can be an e-mail or SMS or anything similar) is a spam or a ham message. This is particularly important because we wouldn't want those spam messages to be shown on the main page as it will make us really hard to read and find important messages. The real-life application of this can be found in our E-Mail app. Most of them have a built-in spam filter out of the box and those E-Mails that are classified as "spam" will only appear in the spam folder. This makes our life much easier to read only important E-Mails.

The goal of this project is that we should be able to classify a message to be a "spam" or "ham" just by parsing the entire message into the algorithm. An example of this is shown below:



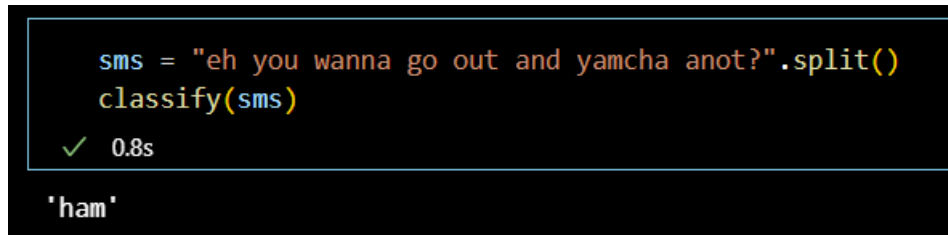
```
sms = "claim your free duck at duckduckgo.com".split()
classify(sms)

[2] ✓ 0.5s
... 'spam'
```

Figure 1.1

In figure 1.1, the sentence "claim your free duck at duckduckgo.com" is being parsed into the classifier that I have built for this project and it is being classified as a 'spam'. This is highly likely to be correct because this is what a spam message will normally look like.

In the next example below, I will show you the message that will normally appear when someone in Singapore/ Malaysia wants to invite his/ her friend to get a drink and chill in a restaurant.



```
sms = "eh you wanna go out and yamcha anot?".split()  
classify(sms)  
✓ 0.8s  
'ham'
```

Figure 1.2

I know it sounds confusing and one may ask “is this even English?”. Well, the story is complicated in Singapore and Malaysia because Malaysians and Singaporeans consist of many different races, and each race speaks different languages. Over a few generations, we are used to mixing a lot of different languages when speaking. The sentence used in figure 1.2 just means “do you want to go out and have a drink?”. When this sentence is being parsed into the classifier, the model predicted it as a “ham”, as what you would expect from a message coming from a friend.

*Disclaimer:

Due to the nature of the Naive Bayes algorithm, all the punctuations in a sentence must be removed before processing. This is not done in the example above, but it shouldn’t be a big problem because in both examples there is only one punctuation in each sentence, which is “.” in “duckduckgo.com” and “?” in “anot?”. For the training and testing of the model, all punctuations are removed from the data set, which I will explain in more detail in the next chapter.

Methodology

This spam filter is built based on the Naive Bayes algorithm. Classifiers that use naive Bayes are a popular statistical technique for email filtering. To understand the Naive Bayes algorithm, we first have to take a look at the Bayes Rule.

Bayes Rule

Definition:

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

Terminology:

$$\textbf{Posterior} = \frac{\textbf{Likelihood} \times \textbf{Prior}}{\textbf{Normalizer}}$$

Naive Bayes Algorithm

The Naive Bayes algorithm relaxes the conditional dependency by assuming conditional independence between the parameters in X given C:

In our case, C is the class that consists of “spam” and “ham”, and X is the class for all the words x.

Definition:

$$P(C|X) = P(C|x_1 \dots x_n) = \frac{P(C)P(x_1 \dots x_n|C)}{P(x_1 \dots x_n)}$$

$$= \frac{P(C)P(x_1|C) \dots P(x_n|C)}{P(x_1 \dots x_n)}$$

$$C \in (SPAM, HAM)$$

In practice:

To get the most likely class, the denominator can be ignored as it is independent of C:

$$C = \textit{argmax}_c P(C) \prod_i P(x_i|C)$$

Bag of Words

It's mentioned earlier that we've assumed conditional independence between the parameters in X given C :

$$P(x_1 \dots x_n | C) = P(x_1 | C) \dots P(x_n | C)$$

With this assumption, we are basically assuming that the order of the word inside of a sentence is not important since they are independent of each other. Because of this, we only need to record all the words occurring in a corpus along with their frequencies.

How does it actually work? Let's consider a corpus that consists of several phrases:

1. CLAIM FREE DUCK
2. THIS PET DUCK IS SO CUTE
3. NO FREE CUTE DUCK

Resulted bag of words for this corpus:

Word	Frequency
CLAIM	1
FREE	2
DUCK	3
THIS	1
PET	1
IS	1
SO	1
CUTE	2
NO	1

Calculations of $P(x_1 | C) \dots P(x_n | C)$, for which x_1, x_2, \dots, x_n are each individual words appearing in the bag of words, are crucial for the Naive Bayes Classifier. It is calculated as follows:

$$P(\text{word} | C) = \frac{\text{frequency of this word appearing in } C}{\text{sum of all the words in } C}$$

To avoid overfitting, I used Laplacian smoothing:

$$P(\text{word} | C) = \frac{\text{frequency of this word appearing in } C + k}{\text{sum of all the words in } C + (k * \text{sum of all vocabularies in all } C)}$$

Where $k > 0$, it acts as a tuning parameter for this model. I used $k = 1$ for this project.

Experimental Results and Discussion

The data set has the following structure:

	Label	SMS
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Figure 2.1

The data set is cleaned (all the letters are converted into small letters, all the punctuations are removed, and all the words are separated from each other in a list). The cleaned data set is then split into training and testing sets with 80:20 ratio. Figure 2.2 shows how does the cleaned data set looks like.

	Label	SMS
0	ham	[go, until, jurong, point, crazy, available, o...
1	ham	[ok, lar, joking, wif, u, oni]
2	spam	[free, entry, in, 2, a, wkly, comp, to, win, f...
3	ham	[u, dun, say, so, early, hor, u, c, already, t...
4	ham	[nah, i, don, t, think, he, goes, to, usf, he,...

Figure 2.2

Results and Metrics

The results of classification on the test data set are as the following:

	Actual Spam	Actual Ham	Sum
Predicted Spam	141	8	149
Predicted Ham	4	960	964
Sum	145	968	1113

Metrics:

Accuracy = 0.9892183

Precision = 0.9463087

Recall = 0.9724138

F_1 = 0.9591837

Conclusions and Prospects

Strengths and Limitations

As shown by the metrics above, my model has a very high prediction accuracy of almost 99%. The precision value, which measures the quantity of the right predictions that the model made, is 95%, which is quite impressive. The recall value is also very high too, which is 97%.

However, this model can only predict SMS spam, which is kind of irrelevant now. The data set I used is also quite old. It dates back to year 2012, which is slightly over 10 years from now. The content and context of the SMS might be irrelevant for today and it might not detect new form of spams that may be developed after year 2012. To improve this model, we need more new data sets to fill the 10 years' worth of information gap, so that it can detect more forms of spam. This spam filter also only works for English (formal English and Singapore/Malaysia style of English). If we want it to work for other languages, a different data set with different languages will be required. The effectiveness of this classifier may be degraded by a technique called "Bayesian poisoning". The addition of random or even carefully selected words that are unlikely to appear in a spam message will cause the spam filter to believe the message to be a "ham"

Conclusions

The Naive Bayes Classifier is a very simple yet powerful algorithm. It can achieve a very high accuracy, precision, recall and F_1 score even without using very complex machine learning algorithm such as a Neural Network.