Spam Classification of SMS Texts

Kyle Morris
Winter 2020
https://github.com/jkylemorris/DSC-Portfolio

We take it for granted – our email inboxes, for the most part, do a great job of classifying incoming emails as either “spam”, or unsolicited written electronic communications, or “ham”, which is probably just a punny variation on “spam”. Spam, of course, gets its name from the Monty Python’s Flying Circus sketch about a restaurant “Lovely spam! Lovely spam.”

However, there is another form of electronic communication many of us utilize on a fairly regular basis that does not quite have the infrastructure that email does and consequently while the volume of spam messages I have received is much lower, their success rate in getting through to me is exponentially higher. I am referring, of course, to Short Message Service messages or SMS Text Messages.

However, this actually presents a golden opportunity for us. Spam (email) generators are quite sophisticated. Between emulating existing company communications, to evading existing spam detectors, spam overlords are in a constant game of cat and mouse with email services, each trying to gain an advantage. As I mentioned, the best I get with spam text messages is sometimes it shows up under “Unknown Sender” along with people who text me without being in my phone’s contacts. Thus, in theory these text messages should be easier to classify as spam.

What other applications can we use our algorithm for once we have compiled it? Well, if we are already identifying text messages at spam, what about utilizing text messaging for customer contact? While working at the Walt Disney World Resort in Florida, our Concierge team had the ability to text and answer questions from guests via text messages. As the numbers never changed, I am certain they had their fair share of unrelated text messages. A filter like our algorithm here would act as a gatekeeper, allowing the legitimate text messages to go through while stopping unsolicited ones. This would free up employees to address legitimate customer contacts instead of having to filter through mountains of spam.

In order to train our model, we will need a collection of text messages. In this case, our dataset has been categorized for us. If we had collected our own text messages, we would have had to do this step ourselves. This makes our project an example of supervised machine learning, as we are training our model based on data that has already been classified by human involvement. In order to use unsupervised machine learning models, we would just use the raw text of the SMS messages and allow the model to group data itself. This is useful in cases where you may not know how many categories your data has and can quickly test different numbers of categories to find the optimum groupings. In this case, however, we know that our data is either spam or not spam. It is a binary choice, basically. It does not make sense that we would have a text message that is only 34% spam – it might be that our model decides that that is the probability of the message being spam, but you do not generally get a text message that is a mixture of legitimate text and spam message unless there is something seriously wrong with your device!

The text messages were collected from a number of different sources. According to the Kaggle dataset page, from which I discovered the data set, the breakdown is as follows:

“A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning

hundreds of web pages.
-> A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.
-> A list of 450 SMS ham messages collected from Caroline Tag's PhD Thesis.
-> Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages."[i]

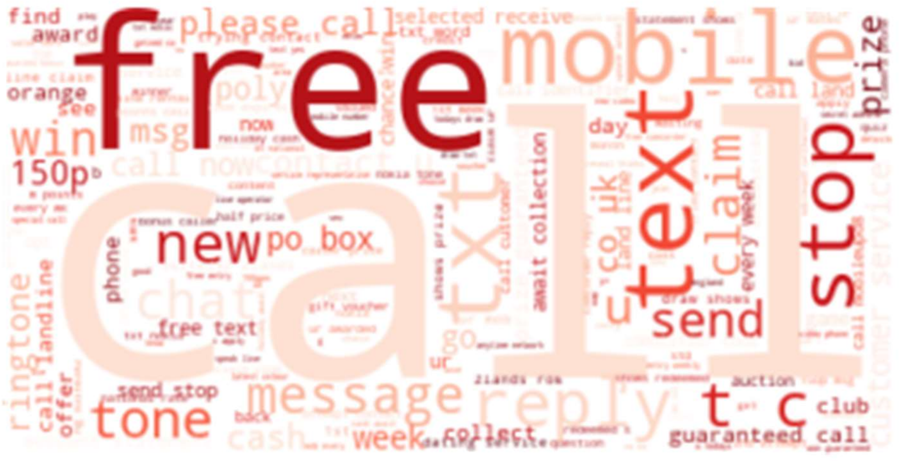
The data consists of two columns, "class", which classifies the message as either "ham" or "spam", and "sms", which is the text of the SMS message. 47 rows have additional information in a third column and 10 in a fourth column, but these are examples of when the SMS message spilled over. These extra rows were cleaned up and folded back into the SMS column, leaving us with just two columns to worry about.

Categorical variables like our "class" column are great for human readability but can be difficult for a computer to work with. One of our first tasks was getting these categorical variables into a form that our models can understand. When you have multiple classifications, it can be tempting to label them 1, 2, 3, etc., but this can cause some categories to be weighted more than others. Is a value of 4 truly twice the magnitude of a value of 2? When your labels are North, South, East, and West, certainly not! This is why a technique called One-Hot Encoding is used in circumstances like this. Fortunately for us, we only had 2 categories so 1 and 0 worked just fine to translate our English categories into numerical levels.

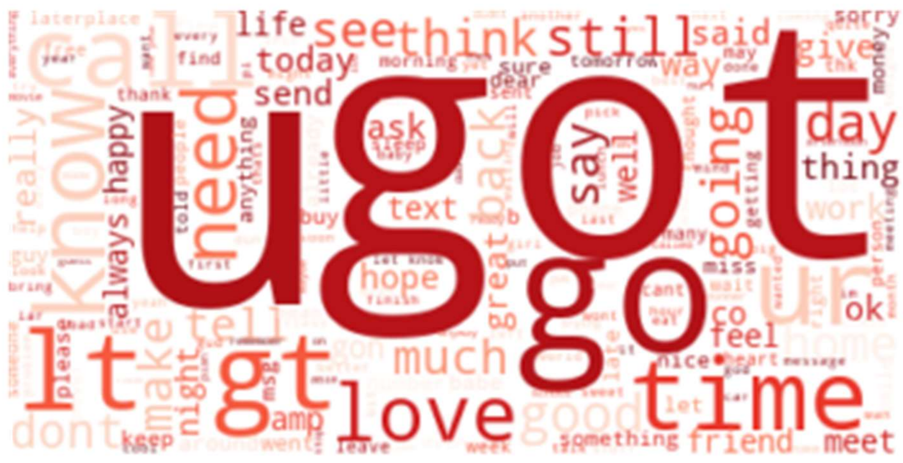
Another part of data cleansing we needed to take care of was we needed to boil things down for our model. The text, in raw form, can be overly complicated. Text messages have symbols and punctuation that can overcomplicate things, and we also need to be aware of a concept called "stopwords". These stop words are common words that are important from a speaking standpoint but can shift focus away from the important part of the sentence. For instance, if you searched for "how to be a better person", if you included the stop words you would end up searching pages for "how", "to", "be", and "a", which might end up distracting from the important part of the sentence i.e., the "better" "person" part.

What we did to prepare our text is removing the punctuation and removing those stop words, leaving a more distilled text for us to analyze. We considered stemming as well but did not go down that route in this case. Stemming is when you boil down the word to its root. It can be difficult when an algorithm has to weight "run, running, ran" down separately when really, it is the same word. So, stemming is when you reduce each word to its root in order to focus on the actual meaning of the words.

With our dataset cleaned, we will want to do some data exploration. One really neat visualization technique we can do is create a word cloud. These were in vogue several years ago. To create them, we split our dataset into a ham dataset and a spam dataset. The word cloud creates a graphic that utilizes the words in each dataset, with the size of the words corresponding to its frequency in the dataset. Thus, more frequently used words are larger. Here is the spam word cloud:



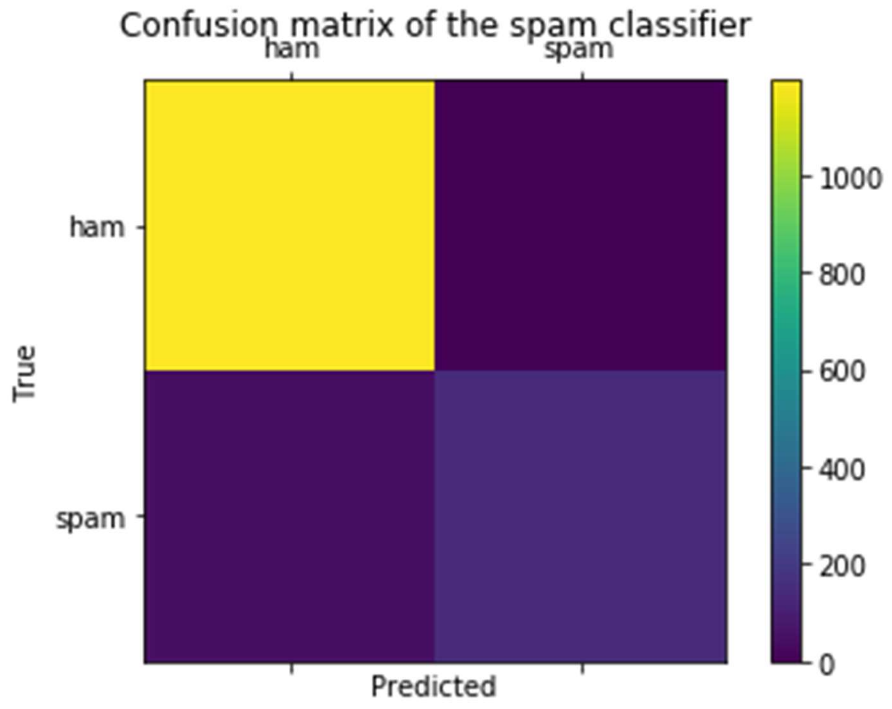
And here is the ham word cloud:



We also looked at the average length of spam text messages versus the average character length of ham text messages. Interestingly enough, the spam text messages were longer on average than the ham text message! In fact, they were almost twice as long – about 72 characters for ham texts, and 139 for spam texts. SMS texts are by nature a medium focused on brevity – perhaps spam artists have learned the longer text messages are more likely to be read. They cannot have a very high conversion rate to begin with.

With our data prepped, it is time to split it into a training set and a test set. This will allow us to train our model on a portion of the data, and then test the resulting models on another subset of the data. This will allow us to figure out how accurate our data is with new data not in the original training set. In this case, we have 4,179 rows of data in our training set and 1,393 rows of data in our testing set.

We utilized two different models to work on our spam classification testing. The first is a multinomial naïve bayes classification algorithm. Upon first run, our Naïve Bayes model shows an accuracy of about 96.3%. That is great! However, accuracy does not tell us the whole story. We want to create what is called a confusion matrix as this is a great way to showcase both how accurate your model is and, when your model is inaccurate, the specific ways it is inaccurate. Here is the confusion matrix for our Naïve Bayes model.



This corresponds to the following matrix: [[1196   0]

[51 146]]. Our model correctly identified 1,196 of our ham messages as ham. It identified 146 of the spam messages as spam. It did not identify any spam messages as ham, and only missed 51 spam messages as ham. Overall, it did a great job of classifying things although a few did slip through the cracks!

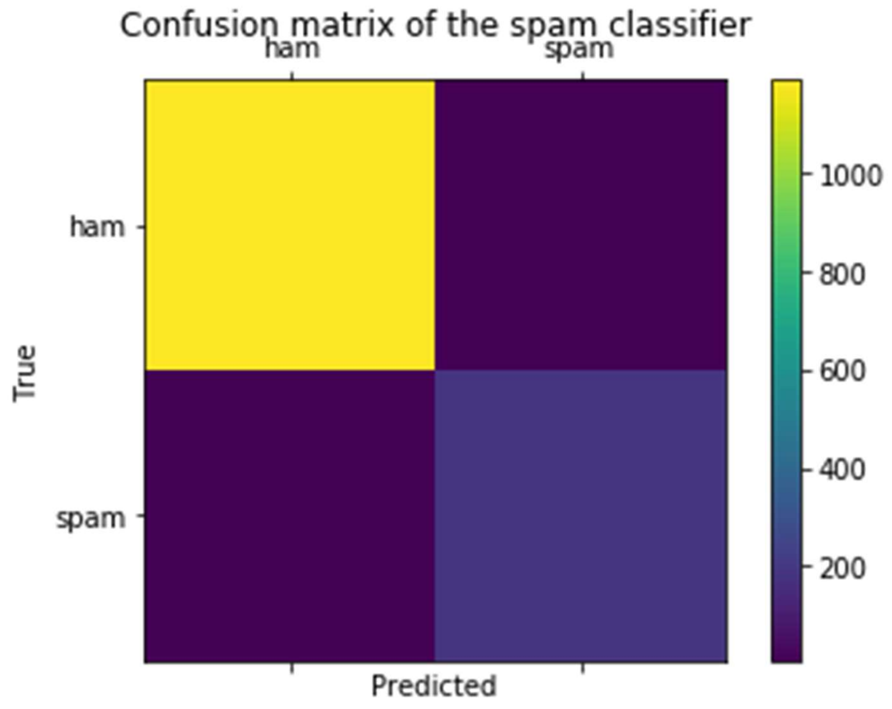
```
       Accuracy Score: 0.9633883704235463
Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1196
           1       1.00      0.74      0.85       197

   micro avg       0.96      0.96      0.96      1393
   macro avg       0.98      0.87      0.92      1393
weighted avg       0.96      0.96      0.96      1393

F-Measure: 0.851
```

The F-Measure for our Naïve Bayes model is .851, which is pretty solid! It is the harmonic mean of the precision and recall, with a maximum of 1 and a minimum of 0. That .851 is pretty high up for the algorithm's ease of implementation.

We were not content to limit ourselves to just one algorithm. We also looked at a classic workhouse, the logistic regression algorithm. Note that this is not the linear regression model as linear regression is more suited to outputting a value whereas logistic regression is focused on classification.

You might be surprised to hear that logistic regression has a solid home in the world of finance, where it has been used to classify prospective lenders in order to identify who is most likely to default. In our case, we are using it to classify our spam messages again.

In this case, our logistic regression by raw accuracy does slightly better than our Naïve Bayes model! It has an accuracy of 96.67%! Let us take a look at our confusion matrix to see how things settle:



The corresponding matrix that created this visualization is [[1189   7]

 [12 185]]. 7 ham messages classified as spam and 12 spam messages classified as ham. This is a lot smaller than the last algorithm, although we did have some spam messages identified as ham, which the last algorithm did not. By the numbers though, only 19 out of 1393 messages were misidentified!

```
Accuracy Score: 0.9863603732950467
Report:
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1196
          1       0.96      0.94      0.95       197

  micro avg       0.99      0.99      0.99      1393
```

```
   macro avg       0.98      0.97      0.97      1393
weighted avg       0.99      0.99      0.99      1393

F-Measure: 0.951
```

The accuracy is much higher for this logistic regression! 98.63% accurate and with an F-Measure of .951 an increase of .1 over the Naïve Bayes classification. We saw that things ended up a lot more accurate for this one and the increase in the F-Score clinches it.

Overall, if I had to choose between the two models I would have to start out with the logistic regression! That being said, we still have some room for improvement but as we saw in the Netflix challenge, the best model is not necessarily a single model but a blended model. The Naïve Bayes model was great at identifying spam messages, never once accidentally classifying them as ham. If we would combine that ability with the logistic regression's general accuracy, we could end up with a more accurate model. Sometimes it is as simple as getting answers from both models and weighting accordingly. If both models agree, we can be reasonably confident in the answer. If they disagree, we can weight the answers according to the model and come to a consensus.

Now that we have our models, are our original plans still viable? Absolutely! I think we would have been happy with an accuracy around 80% for a first run. 80% would not get us to production, but the 98.7% of the logistic regression? We can certainly bring that to market.

If I were to implement this model, instead of automatically tossing out the spam messages I would send them to a different queue. One employee could quickly scan the spam queue, classifying each message as spam or ham. The ham messages would then enter the customer service queue for the next available agent, and the spam messages would be archived.

The great thing is that while our initial dataset trained our messages, they were not representative of the messages our implementation would be receiving. Say, for instance, we were returning to our concierge service from before. Concierge messages would often be about things like booking dining reservations, purchasing tickets, and paying for experiences. It could conceivably confuse our spam filter as spam text messages are often trying to sell you something. Since we would already be classifying messages as we received them, we could feed them back into our model and it would continue to learn and adapt to the messages received.

Overall, I am pleased with how well our model performed, the both of them. Naïve Bayes in particular is popular because it is easy to implement (it was) and it performs well despite how quickly it can be brought online (it did). I look forward to refining these models in the future, starting with experimenting with blended models!

References:

10 Companies Using Machine Learning in Cool Ways. WordStream. https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications.

12 Best Machine Learning Text Classification Tools and Services. Lionbridge AI. (2020, November 6). https://lionbridge.ai/articles/12-best-text-classification-tools-and-services/.

Almeida, T.A., GÃ³mez Hidalgo, J.M., Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.

Chávez, G. (2019, March 6). *Implementing a Naive Bayes classifier for text categorization in Five Steps*. Medium. https://towardsdatascience.com/implementing-a-naive-bayes-classifier-for-text-categorization-in-five-steps-f9192cdd54c3.

Chia, D. (2020, August 18). *Create a SMS spam classifier in python*. https://towardsdatascience.com/create-a-sms-spam-classifier-in-python-b4b015f7404b.

Guide to Text Classification with Machine Learning. MonkeyLearn. https://monkeylearn.com/text-classification/.

How to Recognize and Report Spam Text Messages. Consumer Information. (2021, January 22). https://www.consumer.ftc.gov/articles/how-recognize-and-report-spam-text-messages.

Komando, K. (2019, August 29). *6 clever ways to quash robotext spam messages before it's too late*. USA Today. https://www.usatoday.com/story/tech/columnist/2019/08/29/spam-text-messages-6-clever-ways-stop-them/2141700001/.

Li, S. (2019, February 27). *Building A Logistic Regression in Python, Step by Step*. Medium. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8.

Shaikh, J. (2017, October 30). *Machine Learning, NLP: Text Classification using scikit-learn, python and NLTK.* Medium. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a.

SMS Spam Collection Dataset. Kaggle. (2016, December 2). https://www.kaggle.com/uciml/sms-spam-collection-dataset.

---

[i] *SMS Spam Collection Dataset*. Kaggle. (2016, December 2). https://www.kaggle.com/uciml/sms-spam-collection-dataset.