Hidden Markov Model for SMS Spam Detection

COMPSCI 179 Project

June 11, 2024

By: Aaron Blume, Anderson Lam, Jason Tran

1. Introduction

Encountering spam has been a rampant issue with SMS communication with no sign of stopping anytime soon. Fortunately, there are ways in which we can help automate the filtering of such messages by detecting if a message is spam or not. The UCI machine learning repository contains a widely-used SMS dataset that contains 5,574 SMS messages which are each classified as either "spam" or "ham" (the opposite of spam). Using this dataset, our team opted to represent the dataset as a discrete state hidden Markov model (HMM), where each word gets categorized by the HMM as spam or ham which is then used to categorize the entire message as spam or ham. We trained it using the Baum-Welch algorithm and predicted classifications using the Viterbi algorithm.

2. Resources

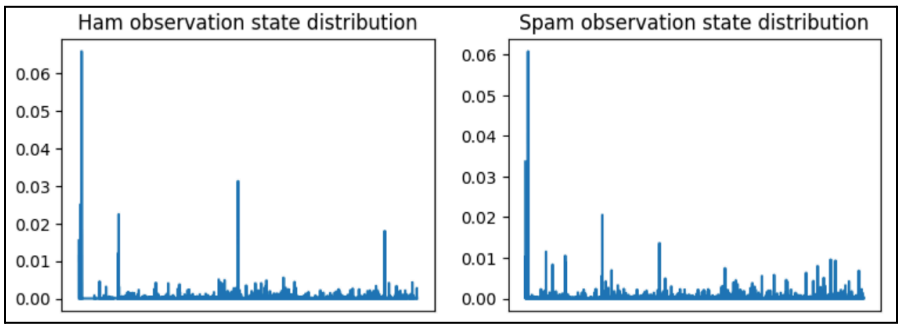
A majority of our work is based on the research done by Tian Xia and Xuemin Chen in their 2020 paper "A Discrete Hidden Markov Model for SMS Spam Detection". Furthermore, a large portion of the code for the Baum-Welch algorithm is borrowed from the 2019 article "Derivation and implementation of Baum Welch Algorithm for Hidden Markov Model", written by Abhisek Jana, with some modifications so it could be applied to our data. These resources can also be found in the "Resources" section.

3. Methodology and Results

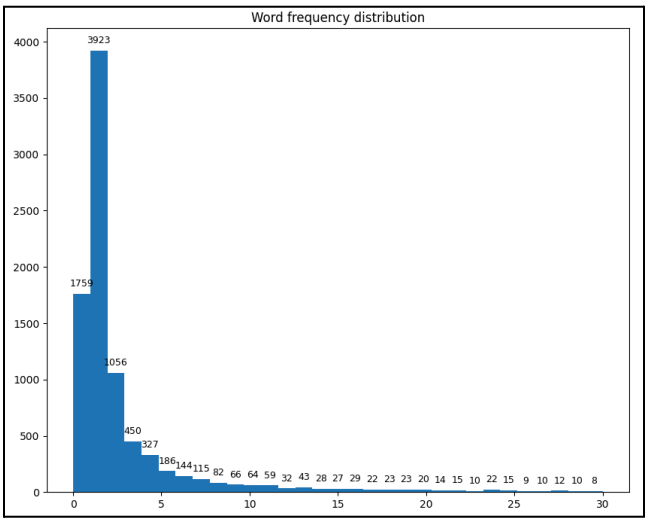
SMS messages contain a wide variety of English words, as every word has multiple forms of itself depending on conjugation, declension, and misspellings. Additionally, when it comes to information retrieval, there are many common English words that add no unique meaning to a message (such as "a" and "the"). These are referred to as stop words. Both these factors tend to add a lot of unnecessary noise to the data. In their preprocessing, Xia and Chen tokenized each SMS message and removed stop words to improve the results returned by their model. We took this a step further by applying lemmatization to each word for added consistency. For example, past tense verbs were changed to their present tense form, and plural nouns were changed to their singular form. In doing so, words of different forms but the same semantic meaning were mapped to the same word. We used Python's NLTK package to tokenize and lemmatize the SMS messages.

Given that the size of this dataset is relatively small, we decided to split it into two-thirds for training and one-third for testing(this ratio is popularly used in other papers). We

then further split the data depending on whether or not it was “ham” or “spam”, while removing any stop words and applying the lemmatizer. After splitting the data, we computed the observation state distribution for the ham and spam datasets to use in our HMM. This involved finding the entire set of processed words and computing the relative frequency of each word in the ham and spam training datasets. To preserve consistency between the 2 separate observation state distributions for spam and ham, we sorted the distributions alphabetically so that each column in the state distribution matrix corresponds to the same word (i.e. the first column is the first alphabetical word, the second column is the second alphabetical word, etc). The image below displays the two distributions’ relative frequencies for each word:



The most notable observation is that a majority of words in the dataset only appeared once. In fact, of the 8,857 unique words identified by our preprocessing steps, nearly 7,000 of them appeared between zero and two times in the entire dataset. Ideally, this would give us a more condensed and consistent observation state distribution since multiple word forms all map to the same word. The following is an image displaying the combined total count of words in both “ham” and “spam”:



The next major step was to train the HMM by optimizing its parameters (namely, the state transition matrix and the emission matrix). The Baum-Welch algorithm is

commonly used to optimize these parameters given a sequence of observations. At a high level, this algorithm is based on the expectation-maximization algorithm in which the HMM parameters are iteratively tuned until they converge at a local maximum of the likelihood. To initialize our model's parameters, we set the initial state and transition matrix such that each state is equally likely (i.e. each binary state has a probability of 0.5). We use the entire training set data to serve as our observation sequence for the training.

As for the model's prediction using the trained parameters, we used the Viterbi algorithm to obtain the most likely sequence of hidden states given an input SMS message. Since this algorithm classifies each word in the message and our goal is to classify the entire SMS message as either ham or spam, we considered the majority classification as the overall message's classification. So, after training our HMM, we were able to achieve the following accuracy measurements on the test dataset.

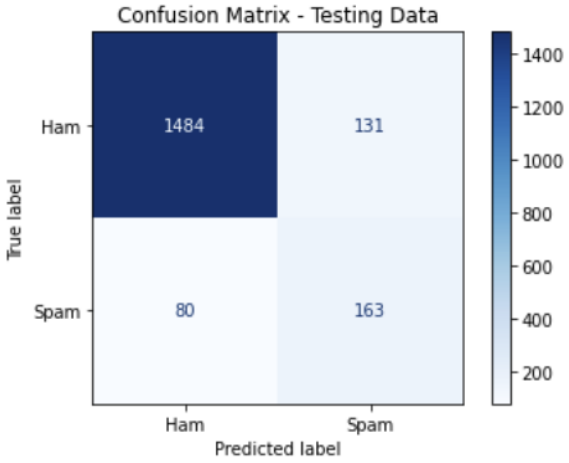
Results	
Accuracy for detecting ham	0.9889
Accuracy for detecting spam	0.5309
Total accuracy	0.9290

This model has a very high false negative rate (identifying spam messages as ham) because when it encounters a word it has never seen before, it defaults to categorizing that word as ham. If we "hack" our way around this by randomly selecting a category for these unknown words (and running 5 iterations), we end up with the results in the following table. At the expense of correctly identifying ham messages, we can correctly identify a larger percentage of spam messages.

Results	
Accuracy for detecting ham	0.9176
Accuracy for detecting spam	0.7119
Total accuracy	0.8907

The tradeoff is relatively substantial, as our model is still able to correctly identify over 90% of ham messages and can now correctly identify over 70% of spam messages. The runtime is reasonable as well, classifying all 1,858 testing SMS messages in under 8

seconds. There are still some obvious limitations to this model, including the small training dataset size and the lack of support for conversational errors and misspellings. In the future, we could employ a more sophisticated language detection model to simplify our observation state distribution as well as run the Baum-Welch algorithm on higher iterations to further optimize the model's parameters.



4. Conclusion

SMS spam detection is a relevant and important topic in today's world, and there have been many efforts to improve its accuracy. Our team developed a discrete HMM which was trained using the Baum-Welch algorithm for 100 iterations. We simplified the complexity of our model by tokenizing each message, removing stop words, and applying lemmatization for consistency. This allowed us to form the observation state distribution of the training dataset which could then be used in the Baum-Welch algorithm alongside an arbitrary initialization of HMM parameters. To predict the classification of each SMS message in the testing dataset, we used the Viterbi algorithm and classified the message based on the majority classification of the words in the message. In the end, we were able to achieve an overall accuracy of 89% which demonstrates the capabilities of our model but also shows room for improvement. Future improvements may include enhanced language preprocessing steps and collecting more SMS message data.

5. References

Almeida, Tiago and Hidalgo, Jos. "SMS Spam Collection." *UCI Machine Learning Repository*. 2012. https://doi.org/10.24432/C5CC84.

Jana, Abhisek. "Derivation and Implementation of Baum Welch Algorithm for Hidden Markov Model." *A Developer Diary*, 20 Feb. 2019, adeveloperdiary.com/data-science/machine-learning/derivation-and-implementation-of-baum-welch-algorithm-for-hidden-markov-model/.

Xia, Tian and Chen, Xuemin. "A Discrete Hidden Markov Model for SMS Spam Detection." *Appl. Sci*. 2020, 10, 5011. https://doi.org/10.3390/app10145011.