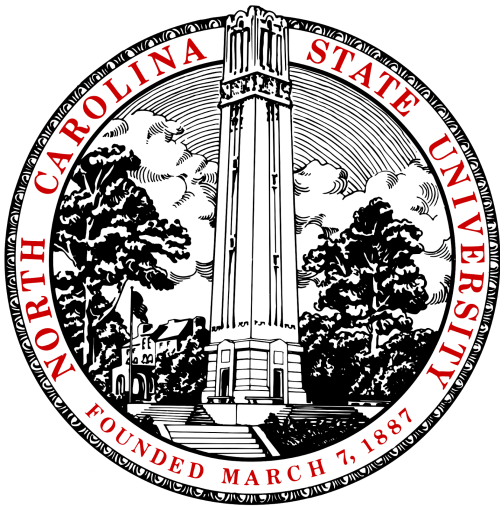
Artificial Intelligence



Data analysis and prediction for SMS spam Collection dataset

Nischal Badarinath Kashyap
nkashya@ncsu.edu
04/23/2020

Problem Statement :

A study by McAfee shows that over 97% of consumers were unable to correctly identify phishing scams when presented with them. As digital communication has evolved so has the sophistication of hackers and scammers. In addition to phony phone calls and email phishing scams, text scams have become more common with the proliferation of smartphones and the emergence of text messages as a part of everyday communication. However, many people could not correctly figure out whether a text is a spam text or a ham text. According to Wikipedia, spam: “the use of electronic messaging systems to send unsolicited bulk messages, especially advertising, indiscriminately.”; ham: “E-mail that is generally desired and isn't considered spam.”. So identify when a text message is spam or ham will benefit people in their daily life.

Abstract :

A machine learning model has been built to classify the given messages into spam or ham. Initially we begin with splitting the data into a training set and testing set. Further we move onto refining the dataset individually for training and testing set by removing the null values if present. The training dataset is prepared for analysis and the test dataset is parsed for prediction. Using these information, we tend to find the accuracy for the model.

Dataset :

We are using the following data set collected from the UCI Machine Learning Repository.

http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

This dataset contains 5574 messages with each message tagged as either spam or ham. The data set contains only two attributes

- Messages Contains messages
- Labels Contains class labels which are either ham or spam.

Here is an example of a single data point in the dataset :

➔ Spam - FreeMsg Hey there darling it's been 3 weeks now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv.

➔ Ham - Even my brother is not like to speak with me. They treat me like aids patient.

Data Pre-processing :

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.

The data is read from a csv file and is stored as a list of lists with each inner list containing the label and the message in string format. This list is of 5574 length as there are 5574 messages. The data is then split into two sets namely, training set and testing set. Each set is then parsed for data preprocessing individually.

The data is parsed for pre processing in order to remove any null values if present.In our dataset, we do not have any data point that contains null values. Therefore we move onto formatting each message into usable format. We initially convert every character in the message into lower case in order to have a unique case sequence throughout the program. The tab spaces are replaced by single spaces and consequently we split the message with respect to space to form a list of words. The first word is always the class label and is separately stored in another variable and the rest of the words are stored in the corresponding data set i:e training dataset or testing dataset.

Evaluation methods :

Naive Bayes with Laplace Smoothing :

In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. In statistics, Laplace Smoothing is a technique to smooth categorical data. Laplace Smoothing is introduced to solve the problem of zero probability. In this project the naive bayes with laplace smoothing is implemented as follows :

The pre processed data acts as an input to the naive bayes. Initially a dictionary of count of each word is created separately for both spam words and ham words. We calculate the probability of messages being spam for the entire dataset and the probability of messages being ham. The test dataset is then parsed to predict the probability of a message being Spam or ham. Initially we consider the probability of a message being spam and ham as 1. Concurrently for a message we calculate the probability of every word being spam and ham and aggregate it into two variables. The formula is as follows :

Probability = Existing Probabilty * [Count_Occurences(spam or ham)* α]/[N + α L]

Where :

Existing Probability Initially 1 and gets updated at each trial

α Constant Value

N Total Number of Spam/Ham Words

L Total Number of Words in the dataset.

These two variables now contain the probability of the message being either a spam or ham. The data is then predicted as being spam or ham based on the value of each probability. The prediction is later validated using the original class labels and the accuracy is calculated.

Results :

In this project since we have enabled the shuffle during data split, the accuracy of the model is swaying between 95% to 97%. The time complexity is highly efficient with O(N) complexity.

Cosine Similarity :

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. A commonly used approach to match similar documents is based on counting the maximum number of common words between the documents. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

In this project, the pre processed data feeds as the input to the cosine prediction function. Here we loop a test message which is in the form of a sentence. The test data point is converted into words using word_tokenize from inbuilt nltk library. The cosine similarity is calculated between every test_data and the data in ham and spam lists individually. Later we aggregate the combined cosine similarity value for spam_data and store it in total1. Simultaneously we aggregate the combined cosine similarity value for ham_data and store it in total2. We classify the data point with respect to the scale of the values in total1 and total2. This gives us the predicted class label. Gradually we check for the accuracy for the model.

Results :

In this project since we have enabled the shuffle during data split, the accuracy of the model is swaying between 82.5% to 92%. Due to three for loops the time complexity is $O(n^3)$ and it takes 50 minutes to provide us the accuracy for the model.

Challenges :

The major challenge I saw in this project was the lesser number of attributes present in the dataset. With only one attribute namely message it was challenging for me to decide on which model to work on. Gradually I chose naive bayes due to its simplicity and accuracy. I also worked on Cosine Similarity in order to explore and understand the working of the model.

The second major challenge was the data pre processing part where we had to split the dataset accordingly and convert it into the desired format as required by each model. For example, Naive Bayes takes in words as inputs while Cosine Similarity had to be fed with Sentences.

The third major challenge was the logic writing. In this project i have not used any inbuilt libraries and hence the project necessitated the complete code to be written with logic. This required a lot of time and dedication from the student's point of view.

The last challenge was in cosine similarity. Although I tried to reduce the time complexity I was unable to continue forward without having $O(n^3)$ time complexity. Trying to reduce the time consumed by the program was a major hurdle.

Acknowledgements :

I would like to show gratitude to Professor Dr Bita Akram and Teaching Assistants for their continuous guidance, insights and providing us with an opportunity to work on this dataset.

References :

- https://en.wikipedia.org/wiki/Additive_smoothing
- https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a
- https://www.machinelearningplus.com/nlp/cosine-similarity/
- https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/cosine/