DISCRETE MATHEMATICS FOR COMPUTING

GROUP PROJECT 2

Bayesian Spam Filters

ie UNIVERSITY

PROFFESSOR: MANUELE LEONELLI

GROUP 4: MAX HEILINGBRUNNER, PABLO OSTOS BOLLMANN, AIKATERINI ORLOVA, NICCOLO' MATTEO BORGATO, WENDY QUARSHIE, JOSEPH GUSS

GROUP PROJECT 2

Table of contents

1. Introduction

In this paper, we will be addressing Bayes Theorem, the in-depth mathematical explanation of it as well as a potential implementation of it as a Spam Filter.
The Bayes' theorem, also known as Bayes' law, named after Thomas Bayes, is a concept in probability theory and statistics that estimates the likelihood of an event based on the knowledge of potential confounding factors. Bayes' Theorem seeks to determine the likelihood that a specific event will occur based on circumstantial evidence. There are various applications of the theorem in different fields, from medicine to law, from machine learning to engineering, and software development. We will present the mathematical concepts behind the law and deep dive into an implementation explanation of the filter itself using an example of a Spam Filter based on Bayes' Theorem.

2. Mathematical explanation

To present the mathematical foundation behind the Bayes Theorem we will use a theoretical example of developing a spam filter ourselves. Let's suppose a certain suspicious message contains the word "offer". Maybe a human being easily identifies this message as spam, but a software detection filter does not "know" this, so all it can do is calculate probabilities. Most of this software uses the next formula, derived from the Bayes Theorem, to determine if a message is spam or not:

$$Pr(S|W) = \frac{Pr(W|S)Pr(s)}{Pr(W)} = \frac{Pr(W|S)Pr(S)}{Pr(W|S)Pr(S) + Pr(W|H)Pr(H)}$$

Where:

- **Pr(W|S)** is the probability that a message is spam, knowing that it contains the word "offer".
- **Pr(S)** is the probability that a message is spam.
- **Pr(W|S)** is the probability that the word "offer" appears in a spam message.
- **Pr(H)** is the probability that a message is NOT spam.
- **Pr(W|H)** is the probability that the word "offer" appears in non-spam messages.

Some statistics show that the probability of a message being spam is **80%** (**Pr(S)**=0.8, **Pr(H)**=0.2), nevertheless, most spam filters set the same probability for a message being spam and non-spam (**Pr(S)**=**Pr(H)**=0.5), which makes the filter unbiased. However, being in possession of empirical data might enable the software to set these ratios more accurately. For this explanation, let's imagine **Pr(S)=Pr(H)** which allows the simplification of the formula:

$$Pr(S|W) = \frac{Pr(W|S)}{Pr(W|S) + Pr(W|H)}$$

The last step would be estimating the probability of the word “offer” appearing in a spam message and in non-spam messages. This is done by identifying the number of words that appear in a set of spam messages (|Spam|) and in a set of non-spam messages (|Non-spam|), and the number of messages that contain the word “offer” in each of the sets (n_{spam}("offer'') and n_{non_spam}("offer")). So we are left with the next expressions:

$$Pr(W|S)_{estimated} = \frac{n_{spam}(”offer”)}{|Spam|}$$

$$Pr(W|H)_{estimated} = \frac{n_{Non-spam}(”offer”)}{|Non-spam|}$$

If the probability that a message is spam, given that it contains the word “offer” (Pr(S | W)) surpasses a certain threshold, the message will be classified as spam.

Combination of individual probabilities

Spam filters are unlikely to be successful in analyzing the probability of a message being spam given that it contains a unique word. Therefore, filtering software combines the individual probabilities of several of these events and this combination derives in the next formula:

$$p = \frac{p_1 p_2 ... p_n}{p_1 p_2 ... p_n + (1 - p + 1)(1 - p_2)...(1 - p_n)}$$

Where:

- **p** is the probability that the message is spam.
- p_1 is the probability $p(S|W_1)$ that a message is spam knowing it contains “$word_1$” (i.e. ”sale”).
- p_2 is the probability $p(S|W_2)$ that a message is spam knowing it contains “$word_2$” (i.e. ”discount”).
- p_n is the probability $p(S|W_n)$ that a message is spam knowing it contains “$word_n$” (i.e. ”promotion”).

If p is lower than the threshold, the message is likely not to be spam, and if it is higher it will likely be spam.

3. Implementation explanation

Before implementing the Bayes Theorem, it is essential to properly train the model that shall be able to predict incoming emails as spam or valid mail. Therefore, it is crucial to provide the model with a balanced dataset of unique keywords from valid emails and keywords from spam mail. Then the model can iterate through the data frame containing spam and valid words and store only unique words in two separate arrays. Hence, the model can calculate the “spamicity” (probability of spam mail) and “hamicity” (probability of valid emails).

To test the model, it is important to **split the data frame into training and testing data**. To avoid any potential word from spam mail not appearing in the application on the testing data or vice-versa, it is vital to set a constraint. This constraint is that all words that are correlated to spam emails must appear at least once in the model, independent of whether they are separated in the training or testing data. Furthermore, the probability of "spamicity" and "hamicity" would be calculated on a 0 to 100% scale. In the end, the model is able to **predict the probability** of "spamicity" and "hamicity" by dividing the total number of spam words/ valid words by the total number of all words in the data frame.

After computing the probability of being not spam, we can then run through a set of un-labeled test emails to create a list of distinct words. This would be followed by removing words that never appeared in our training set as our model has no idea how to evaluate them therefore they would not affect the overall probability of being spam or not. To further clean the test data, we perform **stemming** and remove all non-keywords. This refers to articles and prepositions like: and, the, you, your, etc. It's important to not overdo it and remove words that may be important, however, it is equally important to clean up non-key words so that the classifier can focus on the more important words to optimize the model.

Now that we have computed all of the probabilities and conditional probabilities that are necessary to use the formula for Bayes' law, the model can be applied. The **overall spam probability** for the email will be computed by producing a 'spam' probability for each word in the email and then multiplying all of the individual word probabilities together. This spam filter will deem any email as spam if it has a spam probability >= 0.5. It is therefore clear why this implementation of the Bayes' classifier is considered naive. It is naive as the relationship between neighboring words is not accounted for, but rather **every word is evaluated independently**.

4. Conclusion

The implementation of a naive Bayes spam filter was very common in the early days of spam filtering. However, today advances in NLP and the use of word embeddings, where inter-word semantics are considered, have allowed for far more advanced spam filters. Similar to most attack and defense relationships, the game of filtering and scammers evolving to bypass these filters is very much a cat-and-mouse game that continues to evolve. For example, the word viagra has a very high spam score, and as a result, spammers may choose to spell it as v1agra or v!iagra in order to bypass a basic filter. This then requires the spam filter to evolve to account for this loophole, and then of course the spammers will then innovate to find the next loophole. It is clear that the importance of Bayes' rule cannot be understated, and leveraging Bayes' rule along with additional tools such as NLP and word embeddings can allow for very robust spam filtering.

5. Annex

Total law of probabilities

Suppose that events A_1, A_2,..., A_n, form a partition of a set **E**, this means that events A_j are independent of each other and their union is equal to the set **E**. Now, let **B** be an event where we know all conditional probabilities $Pr(B|A_j)$, then, the total probability of the event **E** can be expressed with the next formula:

$$Pr(B) = \sum_{i=1}^{n} Pr(B|A_i)Pr(A_i)$$

Going back to the spam filter, if we look at the formula to calculate the probability that a message is spam, knowing that it contains a certain word:

$$Pr(S|W) = \frac{Pr(W|S)Pr(s)}{Pr(W)} = \frac{Pr(W|S)Pr(S)}{Pr(W|S)Pr(S) + Pr(W|H)Pr(H)}$$

The fact that $Pr(W) = Pr(W|S)Pr(S) + Pr(W|H)Pr(H)$, is a direct application of the law of total probability. This theorem translates to the probability of the word “offer” appearing in messages is equal to the probability of “offer” appearing in spam messages times the probability of a message being spam, plus, the probability of “offer” appearing in non-spam messages times the probability of a message not being spam.

Bayes Theorem Python Notebook

To provide a hands-on experience, we developed and implemented a program for a Bayesian spam filter. In the attached python notebook, you can test the spam filter based on a dataset, consisting of valid and spam mails. According to our results, the program can predict spam mails with 98.74% accuracy.