CS 267A Final Writeup

Huo, Jianfan
jihuo1116@g.ucla.edu

Du, Yunhao
yudwn010@g.ucla.edu

Wang, Yuchen
ycw509@g.ucla.edu

June 2023

Our final code is in the github with the given link:
`https://github.com/yw509/COM_SCI_267_Final`

Abstract

Our project aims to create an efficient and accurate spam classification model that will significantly enhance the filtering capabilities of messaging platforms. With the exponential growth of online communication, the number of spam messages has become a serious issue, leading to stress and potential safety for users. To address this challenge, we use the power of modeling data with a probabilistic programming language. Our model considers specific words within messages as conditional parameters for classification. We also put some specific words within the message to our model as the conditional parameter for classification. The dataset will include approximately 87% non-spam and 13% spam messages. We split the 80% dataset into training and the 20% dataset into tests to ensure unbiased evaluation. During the training process, we construct a vocabulary table, which will include words commonly associated with spam, such as "call" and "claims." These words are the crucial features for training our model to accurately identify spam messages. After the training is complete, we check if it achieves a predetermined baseline of accuracy. And we also use the Problog to refine the model's predictions and further improve its efficiency.

1 Motivation and Introduction

In the dynamic landscape of today's rapidly evolving information, the alarming prevalence of personal data breaches has reached new heights. Protecting our cherished information against such breaches is of paramount importance. However, it is equally important that once our private information is leaked, we must protect ourselves from dealing with the far-reaching effects. Amidst these perils, the nefarious realms of spam and deceitful information cast an indelible impact on our lives. Though the dangers posed by private information leaks, fraudulent messages have a huge impact on our lives. While sometimes these dangers may seem insignificant, if we don't pay attention to them, the consequences can quickly escalate to terrible levels, including finances, time, and even our lives. Even if we are careful in our lives to avoid becoming victims of these scams, the presence of these massive scams and spam messages can cause us psychological pain. We are keen to reduce the impact of spam on people's lives by implementing a keyword filtering system. During the project, we will develop deeper into the usage of ProbLog, extensively explore the usefulness of the Probabilistic Programming language, and gain a comprehensive understanding of how it benefits data analysis. This groundbreaking solution is designed to protect individuals from the relentless plague of fraudulent messages.

2 Background

For exploring the dataset and creating the spam filter model, we mainly use a Python environment and the Probabilistic Programming language ProbLog. We download the Problog package in our Python environment and use the ProbLog based on the following relevant references which provide the guideline for installation and tutorial:
`https://github.com/ML-KULeuven/problog`

`https://problog.readthedocs.io/en/latest/python.html`

The dataset used was from The UCI Machine Learning Repository. It includes a total of 5574 SMS messages that were collected from free or free research sources on the Internet. You can download the dataset from the following link:
`https://archive.ics.uci.edu/ml/datasets/sms+spam+collection#`

3 Technical Contribution

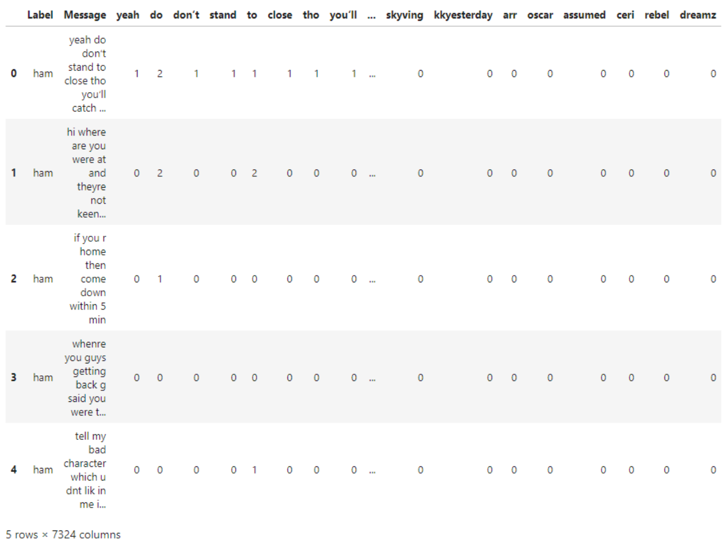
The technical contribution of our project lies in the development of probabilistic programming(based on Problog) and the completion of an efficient and accurate spam classification model. The goal of our project is to address the challenges of today's spam nuisance and provide spam filtering of the message platform. To achieve this, we have considered the following technical aspects:

- Using probabilistic programming languages:
  This is the most basic technical support for our project. In this project, we use the power of probabilistic programming language to help us analyze the data more quickly and effectively. The main programming language we use for modeling is Python. The advantage of using Python is the huge collection of libraries and frameworks for machine learning and natural language processing tasks. Some example libraries we used are pandas, string, and random. The pandas help us to read the csv dataset, create a dataframe, and concatenate different dataframes together. Therefore, we implement a combination of Python and probabilistic programming languages in our project and build a Spam filter with high accuracy.

- Dataset clean and construction:
  As we mentioned in the abstract, we explored a dataset consisting of 87% non-spam and 13% spam. We finished cleaning and classifying the data and obtained 7322 non-duplicate words from all the messages in the training dataset.

  We removed all punctuation, split each sentence based on spaces, and removed all digits from the vocabulary.

  In the beginning, we kept all digits since we think spam messages usually contain phone numbers, money, and time, which are mostly represented by digits. And these digits will help the model to learn the pattern and better detect fake news. However, after we tried to train the model without digits, we found that there was no significant difference between these two situations. Therefore, we decided to remove all digits for more efficient training while maintaining the accuracy rate. This allows us to collect various different information about each message, annotate it correctly and avoid the use of duplicate data for the best performance.

	Label	Message	yeah	do	don't	stand	to	close	tho	you'll	...	skyving	kkyesterday	arr	oscar	assumed	ceri	rebel	dreamz
0	ham	yeah do don't stand to close tho you'll catch ...	1	2	1	1	1	1	1	1	...	0	0	0	0	0	0	0	0
1	ham	hi where are you were at and theyre not keen...	0	2	0	0	2	0	0	0	...	0	0	0	0	0	0	0	0
2	ham	if you r home then come down within 5 min	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	ham	whenre you guys getting back g said you were t...	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	ham	tell my bad character which u dnt lik in me i...	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0

  5 rows × 7324 columns

- Calculation of probabilities and model building:
  We use the Model Counting to calculate some constants $N_{wi|Spam}$, N_{Spam}, $N_{wi|Nonspam}$ and

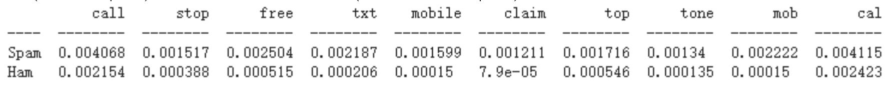
$N_{Nonspam}$ where N_{spam} represents the number of words in all the spam messages so that we can calculate $Pr(wi|pam)$ and $Pr(wi|Nonspam)$ for our model building by using the formula:

$$Pr(wi|Spam) = \frac{N_{wi|Spam}}{N_{Spam}}$$

After our application of the Naive Bayes Algorithm, we have completely solved the difficulties. To improve the accuracy of our model, we also tried to use Laplace Smoothing. We add α that is 1 into our Probabilities function to ensure our accuracy. The formula is:

$$Pr(w_i|Spam) = \frac{N_{wi|Spam} + \alpha}{N_{Spam} + \alpha \cdot N_{Words}}$$

- Constructing spam words table:
  During the training process, we construct a vocabulary table containing some words that are commonly associated with spam, such as "claim", "free" and "call". We determine whether a word is fraud-related based on its probability of appearing in a spam message and then construct a vocabulary table with associated words of spam that have a higher probability of $Pr(Spam|wi)$ compared to $Pr(Nonspam|wi)$.

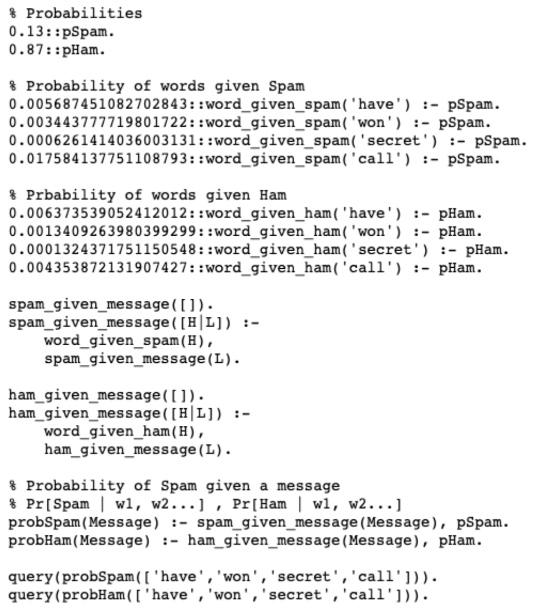


```
         call      stop      free       txt    mobile     claim       top      tone       mob       cal
----  --------  --------  --------  --------  --------  --------  --------  --------  --------  --------
Spam  0.004068  0.001517  0.002504  0.002187  0.001599  0.001211  0.001716  0.00134   0.002222  0.004115
Ham   0.002154  0.000388  0.000515  0.000206  0.00015   7.9e-05   0.000546  0.000135  0.00015   0.002423
```

- Building the spam filter using ProbLog:
  This is the main challenge we have faced in our project. We used the Problog knowledge taught in class and learned outside of class to test our model probabilities and ensure we have achieved the baseline we envisioned. During the project, we discovered that evidence() and query() methods which help a lot when we try to classify a message based on a certain group of words.

  As we learn from the related work, we know ProbLog is mostly written in Python, so this allows us to import ProbLog as a Python package. A ProbLog program can also be composed as a string. We generate a function called probLogClassifier which will generate our ProbLog code as a string and feed it into the Python library called ProblogString. We will get the probability of Spam giving a message and the probability of Nonspam giving a message after calling the two queries probSpam and probHam. The function will finally output the prediction by comparing these two probabilities and the ProbLog code.

  To test all the sentences, we create a for loop outside of Problog to iterate through the entire test dataset and use each message as input to the function to derive probabilities and classify them based on the higher probability.



```
% Probabilities
0.13::pSpam.
0.87::pHam.

% Probability of words given Spam
0.005687451082702843::word_given_spam('have') :- pSpam.
0.003443777719801722::word_given_spam('won') :- pSpam.
0.0006261414036003131::word_given_spam('secret') :- pSpam.
0.017584137751108793::word_given_spam('call') :- pSpam.

% Prbability of words given Ham
0.006373539052412012::word_given_ham('have') :- pHam.
0.0013409263980399299::word_given_ham('won') :- pHam.
0.0001324371751150548::word_given_ham('secret') :- pHam.
0.004353872131907427::word_given_ham('call') :- pHam.

spam_given_message([]).
spam_given_message([H|L]) :-
    word_given_spam(H),
    spam_given_message(L).

ham_given_message([]).
ham_given_message([H|L]) :-
    word_given_ham(H),
    ham_given_message(L).

% Probability of Spam given a message
% Pr[Spam | w1, w2...] , Pr[Ham | w1, w2...]
probSpam(Message) :- spam_given_message(Message), pSpam.
probHam(Message) :- ham_given_message(Message), pHam.

query(probSpam(['have','won','secret','call'])).
query(probHam(['have','won','secret','call'])).
```

By integrating these technical components, our project aims to complete a spam classification model to help people reduce the nuisance of spam in their lives. Data cleaning, the construction of vocabulary tables, calculating probabilities using Naive Bayes Algorithm, and the usage of ProbLog will usefully help us to generate the model, improve its accuracy, and contribute to improving the filtering capability of the information platform.

4 Related Work

In the section on related work, we apply some Machine Learning Techniques, Ensemble Methods, and Evaluation Metrics. We will then go through each of them in turn to describe their usefulness in our projects:

- Naive Bayes Algorithm
  We have used the Naive Bayes Algorithm to calculate the probability that the given message is spam or non-spam given some of the words. The idea is based on the following equations:

  $$Pr(Spam|w_1, w_2, ...) = Pr(Spam) * \prod_{i=1}^{n} Pr(w_i|Spam)$$

  $$Pr(Nonspam|w_1, w_2, ...) = Pr(Spam) * \prod_{i=1}^{n} Pr(w_i|Nonspam)$$

  where wi represents the word in the message sentence.

  To test the accuracy of our model probabilities, we also explore the use of the probabilistic programming language Problog to calculate these probabilities and see if these two methods would produce similar results. Before that, we will use Model Counting to calculate $Pr(wi|Spam)$ and $Pr(wi|Nonspam)$ inside the formula above. We use the following equations:

  $$Pr(w_i|Spam) = \frac{N_{wi|Spam}}{N_{Spam}}$$

  $$Pr(w_i|Nonspam) = \frac{N_{wi|Nonspam}}{N_{Nonspam}}$$

  $N_{wi|Spam}$ represents the number of times the words wi occurs in spam messages

  N_{Spam} represents the total number of words in spam messages

  $N_{wi|Nonspam}$ represents the number of times the words wi occurs in non-spam messages

  $N_{Nonspam}$ represents the total number of words in non-spam messages

  where N_{Words} represents the number of unique words in all the messages of the training dataset.

  On the basis of calculating these probabilities, we get the relationship between these words and scam information. In other words, we find out which words are dependent on scam messages in order to help us more accurately identify the spam message in our training model.

- Laplace Smoothing
  Due to the limitation of our training dataset, it is impossible for us to explore all combinations of a group of words. So, we encounter the zero probability problem when calculating the above equations.
  We learned that Laplace smoothing, a well-established concept widely used in natural language processing and machine learning, can help us solve the above problems. Laplace smoothing, also known as additive smoothing, is a strategy designed to solve the problem of zero probability of unseen events in the training sample, which is mainly used in the Naive Bayes machine learning algorithm. The main idea of this method is to add a small constant to each probability, thereby ensuring that the probability of any possible event is non-zero. Previous research has shown the effectiveness of Laplace smoothing in reducing over-fitting and solving problems with sparse data.
  Therefore, we use Lapace Smoothing to help us address the zero-probability problem and our updated formulas are:

$$Pr(w_i|Spam) = \frac{N_{wi|Spam} + \alpha}{N_{Spam} + \alpha \cdot N_{Words}}$$
$$Pr(w_i|Nonspam) = \frac{N_{wi|Nonspam} + \alpha}{N_{Nonspam} + \alpha \cdot N_{Words}}$$

where α will be a small integer for avoiding the zero probability issue. As α increases, the likelihood probability moves towards uniform distribution, which is 0.5. Most of the time, $\alpha = 1$ is used to remove the problem of zero probability. With this theoretical basis, we have tried the different numbers of the α. When $\alpha = 10$, the accuracy is 94.25%. And when α drops to 1, we got the highest accuracy is 98.83%. This also satisfies the Laplace Smoothing technique.

5 Conclusion

Our project has made significant progress towards achieving our initial aim: developing an efficient and accurate spam classification model using different methods including ProbLog. Our project can identify spam messages in most cases, ensuring users' experience on the platform and addressing potential personal security issues.

Firstly, we compared the accuracy of our model using the Naive Bayes algorithm and the accuracy calculated with the Laplace Smoothing. For the Laplace Smoothing method, we initially set $\alpha = 10$, and there is no significant improvement. The larger the alpha, the closer it will be to the uniform distribution, so we decide to reduce the *alpha* to 1, and our accuracy rate has been greatly improved, from the original 94% to 98.83%. It proves that the Laplace Smoothing truly plays an important role in the Naive Bayes algorithm tasks.

Furthermore, we have also created a spam classification model with the usage of the probabilistic programming language Problog. It implements a combination of Python and ProbLog and it is meaningful that we have learned how to run ProbLog code in Python.

However, there is still some room for further attention and improvements. One of the directions is the feasibility of creating a table of spam-associated words. For the spam-word table, we only have one dataset for it to test in our model. The data is insufficient, so the table is inconclusive. We need more data to further improve the power of the word list. Also, we need to continuously complete and update the words in our spam vocabulary table to ensure that we can keep up with the latest associated spam words in real-time. In future iterations of the spam word, if the spam vocabulary proves impractical, it will be critical for us to address this issue and identify some new ways to handle it. Among the other methods we are currently considering, there are combining feature selection, ensemble methods, or deep learning techniques. We can use these new solutions to capture more relevant features related to spam for spam classification.

In addition, it is critical to continuously update and optimize the model to be adaptive to changing spam messages. We will spend more time on continuously updating and tuning our models to maintain their accuracy and efficiency. We will also think about and try techniques such as online learning. Through those new techniques, our models can continuously learn from new data, ensure first-hand knowledge of the latest spam patterns in real-time, and adapt to emerging spam messages.

In conclusion, our project successfully achieved the expected goal and completed the development of an efficient and accurate spam classification model. However, our future work will focus on the continuous improvement and updating of spam vocabulary table accuracy, exploration of new alternative techniques, such as online learning, and continuous optimization of models. By addressing these issues, we can continuously improve the efficiency and accuracy of our models, ensure users' experience on the platform and address potential personal security issues.

6 Feedback

What went well?

- Clear Object: Our project started with a very clear goal of using probabilistic programming and Problog to create an efficient and accurate spam classification model that would significantly enhance the filtering capabilities of messaging platforms. This clear goal allows us to continue to focus on our projects.

- Usage of probabilistic programming and Problog: By utilizing probabilistic programming languages, we can analyze data efficiently. By using the ProbLog classifier, the accuracy is 98.83% which shows the feasibility ProbLog classifier.
- Consider spam vocabulary and dataset: By classifying spam-related specific words, and constructing a dataset containing 87% non-spam and 13% spam, our model gets the most fundamental and important support.
- Evaluation and Optimization: After the results reached the expected baseline, we discovered the shortcomings of our model through the problog test, and further improved the model and improved its performance.

What would you prefer to be done differently next time?

- Tool selection and use: Due to our relatively short exposure to Problog, we spent a lot of time on testing the results with Problog. We expect that next time we can get in touch with the Problog earlier or use other familiar probabilistic for testing, which can save a lot of time.
- Feasibility assessment: We need to explore the feasibility of creating the spam-related vocabulary earlier in the project and prepare more other alternatives. In our case, after considering the complexity involved, we drop the glossary. However, to better handle such situations in future projects, we need to prepare multiple contingency plans.
- Planning and timeline: Although we planned everyone's work and timeline at the beginning of the project, with the end of the quarter and final approaching, our project complete in the last few days, and some other method that could be explored did not have time to try. We need to keep updating our timeline over time while planning out the tasks next time.
- Data Processing: In order to improve data accuracy, it is essential to allocate sufficient time for data processing. When performing data cleaning, we did not specifically consider numbers as a criterion for identifying spam-associated words. However, it is worth noting that many spam text messages often include a combination of money symbols and numbers, such as $100. This aspect should be taken into consideration for future improvements and refinements.

7 Reference

Items that are cited: We use the website [Pro] together with website [Leu15] for Problog. And we use the dataset from website [AH12]. We also use the article [Jay20] for the Laplace smoothing and the article [Cha22] for the Naive Bayes Algorithm.

References

[AH12] Tiago Almeida and Jos Hidalgo. *SMS Spam Collection.* UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5CC84. 2012.

[Cha22] Nagesh Singh Chauhan. *Naïve Bayes Algorithm: Everything You Need to Know.* April 8, 2022. URL: `https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html`.

[Jay20] Vaibhav Jayaswal. *Laplace smoothing in Naïve Bayes algorithm.* 2020. URL: `https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece`.

[Leu15] KU Leuven. *The Language. Probabilistic Logic Programming.* 2015. URL: `https://dtai.cs.kuleuven.be/problog/#`.

[Pro] Problog. *Using ProbLog from Python.* URL: `https://problog.readthedocs.io/en/latest/python.html`.