Filtering Spam with Multinomial Naive Bayes

Kevin Thompson

October 5, 2022

1 Introduction

Spam in communications irritate end users and pose a security risk for companies. Employees who do not realize they are interacting with spam may, for example, mindlessly click on attachments within the email and infect their computer with ransomware or other types of malware. An automated solution that filters out spam is therefore a necessary and highly pro-social technique. Since we are only concerned with classifying messages as "spam or "not spam", the spam filtering problem can be reduced to a binary classification problem, which is one of the most well-understood machine learning problems in artifical intelligence.

I solve this problem by implementing an email parsing program that efficiently converts the the great diversity of email files into a uniform data representation and then fit a multinomial naive bayes algorithm using the provided text data.

2 Methodology

The solution is comprised of the following parts:

1. An email parsing engine dynamically assigns parsing strategies to different email formats in an efficient manner. The engine provides a vocabulary and a pandas dataframe.
2. A count vectorizer from the Scikit-learn Package.
3. A grid search crossvalidation object that searches for the optimal smoothing hyperparameter (α) in the family of Multinomial Naive Bayes models.
4. The model with the highest f1-score produced by the crossvalidation object is evaluated on test data to assess its performance.

2.1 Email Parsing Engine

The parsing engine takes a directory of email files and dynamically assigns a parsing strategy to each file at runtime. The files are parsed into an intermediate representation defined by the class "Email" that is stored in an "EmailDatabase" object. The Engine then takes the email objects and converts them into a vocabulary and a dataframe. The vocabulary is bi-directional dictionary that serves as an encoder and decoder for situations in which an alternative encoder is not available. For the naive bayes model used in this project, the scikit-learn "CountVectorizer" is used instead since bayes's rule relies on counts in the discrete setting.

The Universal Modeling Language (UML) class diagram representig the email preprocessing program is shown in Figure 1 and the code that implements this program can be found in Appendix A1.

2.2 Multinomial Naive Bayes

3 Results