INFO371 Problem Set: Bayes-Theorem based Spam Filter

April 28, 2022

Introduction

In this problem set you will use Bayes Theorem to categorize emails from Ling-Spam corpus into spam and non-spam. Using a single-word-based Bayes approach does not give good results, but this problem set serves as a preparatory work for understanding the Naive Bayes later.

Ling-Spam emails

The corpus contains ~2700 emails from academic accounts talking about conferences, deadlines, papers etc, and peppered with wonderful offers of viagra, lottery millions and similar spam messages. The emails have been converted into a csv file that contains three variables:
spam true or false, this email is spam
files the original file name for this email (not needed in this PS).
message the content of the email in a single line

1 (5pt) Explore and clean the data

First, let's load data and take a closer look at it.

1. (2pt) Load the *lingspam-emails.csv.bz2* dataset.

   Browse a handful of emails, both spam and non-spam ones, to see what kind of text we are working with here.

   Hint: check out *textwrap* module to print long strings on multiple lines.

2. (3pt) Ensure the data is clean: remove all cases with missing *spam* and empty *message* field. We do not care about the file names.

---

2 (15pt) Create Document-term matrix (DTM)

The first serious step is to create the document-term matrix (DTM). This is simply numeric indicators for selected words: does this email contain the word (1) or not (0). But before we get there, we have to decide the words.

1. (2pt) Choose ~ 10 words which might be good to distinguish between spam/non-spam. Use these four: *viagra*, *deadline*, *million*, and *and*. Choose more words yourself (you may want to return here and reconsider your choice later).

2. (10pt) Convert your messages into DTM. We do not use the full 60k-words DTM here but only a baby-DTM of the 10 words you picked above. You may add the DTM columns to the original data frame, or keep those in a separate structure.

   Creating the DTM involves finding whether the word is contained in the message for all emails in data. You can loop over emails and check each one individually, but pandas string methods make life much easier. You will want to do case-insensitive matching, checking for both upper and lower case. You may consider something like this:

```
for w in list_of_words:
    emails[w] = emails.message.str.lower().str.contains(w)
```

   It is more intuitive to work with your data if you convert the logical values returned by `contains` to numbers.

3. (3pt) Split your work data (i.e. the DTM) and target (the spam indicator) into training and validation chunks (80/20 is a good split).

---

3 (80pt) Estimate and validate

Good. Now you are ready with the preparatory work and it's time to dive into the real thing. Let's rehearse the Bayes theorem here again. We want to estimate the probability that an email is spam, given it contains a certain word:

$$\Pr(category = \mathsf{S}|w = 1) = \frac{\Pr(w = 1|category = \mathsf{S})\ \Pr(category = \mathsf{S})}{\Pr(w = 1)}. \tag{1}$$

In order to compute this probability, we need to calculate three other probabilities:

$\Pr(category = \mathsf{S})$	Probability of spam in data
	we also need $\Pr(category = \mathsf{NS})$ for non-spam
$\Pr(w = 1)$	Probability the word is seen in messages
	We also need $\Pr(w = 0)$, probability the word is not seen in messages
$\Pr(w = 1|category = \mathsf{S})$	probability the word is seen in messages that are spam
	We also need $\Pr(w = 1|category = \mathsf{NS})$,
	probability the word is seen in messages that are not spam

...but it turns out we are still not done with preparations. Namely, you need to compute quite a few different probabilities below, including $\Pr(category = \mathsf{S})$, $\Pr(category = \mathsf{NS})$, $\Pr(w = 1)$, $\Pr(w = 0)$, $\Pr(w = 1|category = \mathsf{S})$, $\Pr(w = 0|category = \mathsf{S})$, $\Pr(w = 1|category = \mathsf{NS})$, $\Pr(w = 0|category = \mathsf{NS})$.

1. (2pt) Design a scheme for your variable names that describes these probabilities so that a) you understand what they mean; and b) the others (including your grader) will understand those!

   Hint: you may get some ideas from the Python notes, Section 2.3 *Base Language*.

The first task is to compute these probabilities. Use only training data for this task.

2. (4pt) Compute the priors, the unconditional probabilities for an email being spam and non-spam, $\Pr(category = \mathsf{S})$ and $\Pr(category = \mathsf{NS})$. These probabilities are based on the *spam* variable alone, not on the text.

The next tasks involve computing the following probabilities for each word out of the list of 10 you picked above, I recommend to avoid unneccessary complexity and just to write a loop over the words, compute the answers 3–8, and print the word and the corresponding results there.

3. (4pt) For each word w, compute the normalizers, $\Pr(w = 1)$ and $\Pr(w = 0)$.

   Hint: this is $\Pr(\textit{million} = 1) = 0.0484$. But note this value (and the following hints) depends on your random training/validation split!

4. (7pt) For each word w, compute $\Pr(w = 1|\textit{category} = \mathsf{S})$ and $\Pr(w = 1|\textit{category} = \mathsf{NS})$. These probabilities are based on both the *spam*-variable and on the DTM component that corresponds to the word w.

   Hint: $\Pr(\textit{million} = 1|\textit{category} = \mathsf{S}) = 0.252$

5. (5pt) Finally, compute the probabilities of interest, $\Pr(\textit{category} = \mathsf{S}|w = 1)$ and $\Pr(\textit{category} = \mathsf{S}|w = 0)$. Compute this value using Bayes theorem, not directly by counting!

   For the check, you may also compute $\Pr(\textit{category} = \mathsf{NS}|w = 1)$ and $\Pr(\textit{category} = \mathsf{NS}|w = 0)$

   Hint: $\Pr(\textit{category} = \mathsf{S}|\textit{million} = 1) = 0.843$. But note this number depends on your random testing-validation split!

6. (6pt) Which of these probabilities have to sum to one? (E.g. $\Pr(\textit{category} = 1) + \Pr(\textit{category} = 0) = 1$.) Which ones do not? Explain!

Now we are done with the estimator. Your fitted model is completely described by these probabilities. Let's now turn to prediction, using your validation data. Note that we are still inside the loop over each word w!

7. (8pt) For each email in your validation set, predict whether it is predicted to be spam or non-spam. Hint: you should check if it contains the word w and use the appropriate probability, $\Pr(\textit{category} = \mathsf{S}|w = 1)$ or $\Pr(\textit{category} = \mathsf{S}|w = 0)$.

8. (5pt) Print the resulting confusion matrix and compute accuracy, precision and recall.

9. (5pt) Which steps above constitute model training? In which steps do you use trained model? What is a *trained model* in this case? Explain!

   Hint: a trained model is all you need to make predictions.

Now it is time to look at your results a little bit closer.

10. (4pt) Comment the overall performance of the model–how do accuracy, precision and recall look like?

11. (8pt) Explain why do you see very low recall while the other indicators do not look that bad.

12. (8pt) Explain why some words work well and others not:

    (a) why does "million" improve accuracy?

    (b) why does "viagra" not work?

    (c) why does "deadline" not work?

    (d) why does "and" not work?

    Hint: You may just see where in which emails these words occur, and how frequently. These are all different reasons!

Finally, let's add Laplace smoothing to this model. One can imagine Laplace smoothing as two additional "ghost" observations, one spam and one non-spam. Both of these ghost observations contain every single word in our DTM. See also Lecture Notes, Ch 7.3.2 "Smoothing: how to compute probabilities with too few data", page 263.

Laplace smoothing does not add anything here but it is is a crucial tool when we move to Naive Bayes later.

13. (5pt) Add such smoothing to the model. You can either literally add two such lines of data, or alternatively manipulate the way you compute the probabilities.

14. (5pt) Repeat the tasks above: compute the probabilities, do predictions, compute the accuracy, precision, recall for all words.

15. (4pt) Comment on the results. Does smoothing improve the overall performance?
