DAT565/DIT407
Introduction to Data Science and AI

Assignment 3

Deadline: 2024-09-26 08:00

Problem 1: Spam & Ham

The three files `20021010_easy_ham.tar.bz2`, `20021010_hard_ham.tar.bz2`, and `20021010_spam.tar.bz2` (available on Canvas) contain three sets of emails: easy *ham*, hard ham, and *spam* [1]. "Ham" refers to the benign, good emails that we want to read. "Spam", on the other hand, refers to unsolicited bulk email, also known as *junk*. We are going to adopt a bag of words approach to train a *Naïve Bayesian Classifier* to tell the two classes of email apart.

A. Data exploration

Explore the three datasets. Use your judgment as a human being to describe what makes you able to tell spam apart from ham. Also explore the differences between the two classes of ham (easy and hard ham). What makes them different? Write approximately one paragraph.

B. Data splitting

Perform an appropriate train-test split on the datasets. We will use the training sets to train a classifier, and evaluate its performance against the test sets.

Problem 2: Preprocessing

A *bag of words* maps text documents into vectors of word counts. That is, each word is assigned a token number, and we count the number of occurrences of each unique word in each document. The vector then consists of these counts.

Use the `CountVectorizer` class in Scikit-Learn[1] to convert the emails to vectors (or rather the set of emails into a matrix). Acquaint yourself with the documentation of the class because it is nontrivial to use (specifically understand what the methods `fit`, `transform`, and `fit_transform` do and how to use them in this setting). In your report, explain succinctly what you did. You don't have to separate the headers from the body of the email; processing the entire email is ok.

Problem 3: Easy Ham

We are going to try out two different Naïve Bayesian Classifiers:

- The Multinomial Naive Bayesian Classifier[2], and

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html`

- The Bernoulli Naive Bayesian Classifier[3].

For this problem, use the set *easy ham* as your ham. Use the training sets of ham and spam to train an instance of both classifiers. Then, evaluate the classifier against the test set.

In your report, report the *accuracy*, *precision*, and *recall* of your classifiers, that is, six numbers. Also provide the *confusion matrix* for both classifiers (2×2 matrix where positive/negative predictions are the columns, and actual positive/negative values as rows). The count of true/false positives and true/false negatives should be easily readable in the matrix, together with the marginal sums[4].

Problem 4: Hard Ham

Repeat the experiment of the previous problem, but use hard ham instead. Report the same values. Discuss the differences in results.

Hints

- The train-test split is important to carry out correctly, as doing it correctly will be required for a passing submission; the sets should be disjoint.
- Tarballs (`tar.gz` files) can be read using the `tarfile` module[5] but this is not required; you can also extract the tarballs and read files individually.
- Some of the files are encoded in plain 7-bit ASCII, some in ISO-8859-1, and some in UTF-8, set the encoding appropriately; using the wrong encoding is probably not consequential because the content is mostly in English, but this can cause trouble that can be addressed in multiple ways (e.g., infer the coding form the `charset` attribute in the header, or just make a guess and catch the exception if your initial guess fails and try some other coding in that case).
- The `open` function supports the `encoding` parameter but if you don't know the correct encoding, you can open the file in *binary* mode (`rb`) and read it as a byte-stream; you can then manually `decode` the byte stream into a string using an appropriate character set.
- Read scikit-learn documentation carefully: What does `fit` do, what does `transform` do, and what does `predict` do?

Returning your report

Write a report, typeset in LaTeX, that answers *all* questions above. Include all your Python code in your report as an appendix, preferably using the `listings` package. Your report should be legible even without having a look at your code.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html`

[4] Sums of columns and rows

[5] `https://docs.python.org/3/library/tarfile.html`

If you refer to outside sources, remember to add an appropriate literature reference (including websites) in references by `\cite`ing the references. It is recommended that you use the package `biblatex` to manage citations.

Place your figures in numbered `figure` environments, with descriptive captions and `\ref` to the figures in your discussion. Likewise, place your tables in numbered `table` environments with desciprive captions and `\ref` to the tables in your discussion.

After grading, you will be given another attempt to revise your report according to TA comments if it is not considered acceptable.

The deadline is *hard*. Late submissions will not be read at all and are considered failed. This means you will not get any feedback for the first round and the submission is considered a revision; there will be no third attempt, so if a late submission is failed, you will need to participate in a later iteration of the course for a re-attempt.

References

[1] Justin Mason. *Spam Assassin public mail corpus.* Retrieved 2023-11-10. 2004. URL: `https://spamassassin.apache.org/old/publiccorpus/`.