Group Assignment #2 (150pts): CSCI 404/573

Out: Jan 29, 2014

Due: Feb 11 midnight (11:59pm), online submission

Submission

1. All files should be submitted to Canvas. Only one of your team members needs to submit on behalf of the team. Indicate clearly in your submission the team members. You can submit multiple times, but please have the same team member resubmit all required files each time.
2. Implementation: Any programming language you are familiar with (Python is strongly recommended for NLP tasks, if you do not have preference).
3. Submit the **assn2.zip** file that contains 2 folders (q1 and q2). For each question, submit your programs (if any) and a report (**report.** txt, .doc, or .pdf) of what you did and all the various assumptions you made, including the result of running your code, and discuss the results if necessary.
4. (**) for graduate students only.
5. Readings: You should review your notes from class on N-gram models and naïve Bayes classifiers. Please also feel free to read related articles online.

Question 1: N-Gram Models

N-Gram Models provide a means of probabilistic modeling of a language. For this assignment, you are provided a training dataset and a test data set.

Part 1: Using the training data you are to build N-gram models (50 pts)

a) (10 pts) Extract the vocabulary and include a start and end symbol in your vocabulary.
b) Use add-1/Laplace smoothing to create a bi-gram and a tri-gram language model, and do the following tasks:
    a) (15 pts) Randomly generate 50 sequences from these models with likelihood according to the models. Attach the probability (in log-space) to each sentence.
    b) (10 pts) Compute the perplexity of the two models using the test corpus.
    c) (15 pts) Evaluate the models by determining how well they predict a missing word in a sentence. For this, randomly pick a set of at least 30 sentences in which you remove a single word and then determining whether each of the two models correctly completes the sentence when choosing the maximum likelihood sentence completion (i.e. the word that has the highest likelihood given the respective n-gram model). Briefly discuss the result.

Question 2: Text Categorization using Naïve Bayes (100pts)

In this exercise, you will use Naïve Bayes to classify email messages into spam and nonspam groups.

Step 1: Familiarize the Data Set

The data set is a preprocessed subset of the Ling-Spam Dataset that is based on 960 real email messages from a linguistics mailing list.

The dataset is split into two subsets: a 700-email subset for training and a 260-email subset for testing. Each of the training and testing subsets contain 50% spam messages and 50% nonspam messages. Additionally, the emails have been preprocessed in the following ways:

1. Stop word removal: Certain words like "and," "the," and "of," are very common in all English sentences and are not very meaningful in deciding spam/nonspam status, so these words have been removed from the emails.

2. Lemmatization: Words that have the same meaning but different endings have been adjusted so that they all have the same form. For example, "include", "includes," and "included," would all be represented as "include." All words in the email body have also been converted to lower case.

3. Removal of non-words: Numbers and punctuation have both been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.

As you can discover from browsing these messages, preprocessing has left occasional word fragments and nonwords. In the end, though, these details do not matter so much in our implementation.

Step 2: Review the naïve Bayes Classifier

To classify the email messages, we use a multinomial Naïve Bayes model (where the word frequency is used in feature vectors, instead of a Boolean variable that indicates the word occurs or not). Review the notes, including the smoothing method and the evaluation measures.

Step 3: Implementing Naïve Bayes

Feature Selection: Here are some guidelines that will help you to generate your own features.

Data contents

The data pack you downloaded contains 4 folders.

a. The folders "nonspam-train" and "spam-train" contain the preprocessed emails you will use for training. They each have 350 emails.

b. The folders "nonspam-train" and "nonspam-test" constitute the test set containing 130 spam and 130 nonspam emails. These are the documents you will make predictions on. Notice that even though the separate folders tell you the correct labeling, you should make your predictions on all the test documents without this knowledge. After you make your predictions, you can use the correct labeling to check whether your classifications were correct.

Dictionary/Vocabulary

You will need to generate a dictionary for your model. There is more than one way to do this, but an easy method is to count the occurrences of all words that appear in the emails and choose your dictionary to be the most frequent words. If you want your results to match ours exactly, you are suggested to pick the dictionary to be the 2500 most frequent words.

To check that you have done this correctly, here are the 5 most common words you will find, along with their counts.

1. email 2172
2. address 1650
3. order 1649
4. language 1543
5. report 1384

Remember to take the counts over all of the emails: spam, nonspam, training set, testing set.

Feature generation

Once you have the dictionary, you will need to represent your documents as feature vectors over the space of the dictionary words. One possible way to do so:

1. For each document, keep track of the dictionary words that appear, along with the count of the number of occurrences.

2. Produce a feature file where each line of the file is a triplet of (docID, wordID, count). In the triplet, docID is an integer referring to the email, wordID is an integer referring to a word in the dictionary, and count is the number of occurrences of that word. For example, here are the first five entries of a training feature file we produced (the lines are sorted by docID, then by wordID):

1 19 2
1 45 1
1 50 1
1 75 1
1 85 1

In this snippet, Document 1 refers to the first document in the "nonspam-train" folder, "3-380msg4.txt." Our dictionary is ordered by the popularity of the words across all documents, so a wordID of 19 refers to the 19th most common word.

Training and testing

Finally, you will need to train your model on the training set and predict the spam/nonspam classification on the test set. You are suggested to have separate scripts for training and testing. That way, after you trained your model (that is, after calculated all the parameters of the model), you can run the testing independently as long as the parameters are stored.

Calculate the parameters (the necessary probabilities) from the training data.
Using the model parameters you obtained from training, classify each test document as spam or non-spam.

Note: Be sure you work with log probabilities.

Evaluation

Draw the 2x2 table to report the number of false positive, true positives, false negatives, and true negatives.

Report the precision, recall and f-score of your system.

(**) Smaller Training sets:

Let's see how the classification performance changes when you train on smaller training sets, but test on the same test set as before. So far you have been working with a 960-document training set. You will now modify your program to train on 50, 100, and 400 documents (the spam to nonspam ratio will still be one-to-one).

Select email subsets of 50, 100, and 400, keeping each subset 50% spam and 50% nonspam. For each of these subsets, generate the training features as you did before and train your model. Then, test your model on the 260-document test set and record your classification performance.