Bonjour and Buenos Dias

Overview

In this project you will create a tool to predict what language a document is written in given a sample set of known documents. The tool will be able to use various techniques to make guesses. The basis for each of the techniques will be looking at frequencies of trigrams. A trigram is a three character subsequence of the document, see examples below. Your program will be limited to a short list of languages and each language will have a small set of training data (documents for which the language is known).

Design

You will have a great deal of flexibility with regard to design. You will need to keep track of a lot of data and make many computations on that data. You will probably use both dictionaries and lists. Be sure to keep your functions small, they should do one thing and do it well. You will need to keep track of the trigram frequencies for a specific document. You will also need to keep track of data associated at the language level. Since a language would have multiple known documents, these could be stored as a list of trigram frequencies. As always main should be short, your code should readable and well commented.

Counting Trigrams

In this project you will only be working with ASCII characters.

Our trigrams will not include numbers or punctuation, you will need to remove them from the input documents. Also, all sequences of whitespace will be replaced with a single space. For example two spaces become one space, two enters or returns become one single space. All uppercase letters will be replaced with lowercase letters.

So for example, the string “In this project” breaks down into:

```
“in ” “n t”, “ th”, “thi”, “his”, “is ”, “s p”, “ pr”, “pro”, “roj ”,
“oje”, “jec”, “ect”
```

Given a document you will need to record the frequencies of all possible trigrams that occur in that document. This should be normalized by the size of the document. To do this you will store individual counts for each trigram and divide (normalize) by the total number of trigrams in the document. It is better to represent the document as a single trigram vector or dictionary keeping counts and appropriate data.

Functionality

Python has a great deal of built in functions regarding strings, there are methods to convert to lower case and to determine if the string is alphanumeric. You will need to make use of these. Unlike your previous programs, this program will take in input at the program level and give output at the program level. So when you run the program it would look like:

```
python predict-lang.py input.txt output.txt
```

Input

The input to your program will be a file of filenames and their associated language something like:

```
French ftest1.txt
French ftest2.txt
Dutch nl1.txt
Croatian hr2.txt
Italian itest1.txt
German gt.txt
French ftest3.txt
Unknown ut1.txt
English eng1.txt
Unknown utest3.txt
Spanish st3.txt
```

In the above you will have the name of the language if known and the filename. You can assume that all files will be in the same directory as the program. The name “Unknown” refers to a document that should be tested. Your program should output a file containing the language scores for each Unknown file in order of likelihood:

Output

The output of your program will be a file of filenames and their associated language something like:

```
ut1
        French xx
        Spanish xx
        Italian xx
        English xx


utest3.txt
        German xx
        English xx
        Spanish xx
```

For every known language you will score the unknown document. You will output the guess of the language in order of most likely language. Python has a sort function built in. You will also output the score (xx above). This number will depend on the metric you use to score your language.

You will need to run statistics to compare trigram vectors to each other to determine the language of a document. This is detailed below. Your code should be written so that different techniques can be easily adopted. Namely, we will begin by computing the similarity measure between two vectors using a particular math formula. Your code design should be such that a different formula can easily replace this formula. You should work on the project without the measure first.

Test Files:

We will provide some test files for you to use. You should copy them into your directory and you should play with making files unknown. You are free to add your own test files as well.

Similarity Metric:

There are many ways to classify data. You will learn more about these different techniques in your machine learning class. Two ways are easily implemented for this purposes. The first is a Naive Bayes Classifier. This is a probabilistic model, which uses likelihood and evidence to determine how to classify.

The method everyone is required to implement in their classifier is one using cosine similarity.

We set up the trigrams as a vector. That is, for the ~27,000 possible trigrams you will have a frequency or weight. To compare how close two vectors are you can calculate how close they are with regard to their angle. Recall that the cosine of 0 is 1. Therefore the cosine of a small angle would be 1. So similar documents will be closer to 1.

To calculate the “cosine” we will use the following formula:

$$\frac{\sum_{i=0}^{n} (A_i \times B_i)}{\sqrt{\sum_{i=0}^{n} (A_i)^2} \times \sqrt{\sum_{i=0}^{n} (B_i)^2}}$$

Here A_i is the ith trigram frequency in document A. Recall, that your frequencies will need to be normalized, they should be a percent. You can keep statistics on the overall closeness of documents for a language if that helps in your design. Namely, feel free to make B_i above refer to a language instead of a document.