"The Most Probable Song Title"

Programming Lab 1

Concepts of Programming Languages
CSCI305, Spring 2013

Due: February 20, 2013 at 11:59 pm

Perl

For this lab you will use Perl. Perl is installed on all machines in the lab. Perl should already be installed on OSX and on most Linux systems. To check your version, type `perl -v` at the terminal.

Window users will need to install Perl and have their choice between Active Perl or Strawberry Perl.

Dataset

This lab will make use of a dataset containing a million song titles. This dataset is used in various machine learning experiments and is provided by the Laboratory for the Recognition and Organization of Speech and Audio at Columbia University.

You will need to download the following file:

`http://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/unique_tracks.txt`

This file contains one millions lines and weighs in at 82 MB. You should probably avoid viewing it in a web browser.

File Template

You should download the lab file template at `http://nisl.cs.montana.edu/~pdonnelly/CSCI305/docs/template.lab1.pl`.

First, rename this file to `[LastName].[FirstName].lab1.pl` where `[LastName]` and `[FirstName]` are your last and first names, respectively. Do not include the brackets `[ ]` in your file name. Secondly, edit the header comments in the file to reflect your name. Also edit the variable declaration `my $name` so the program will print your name when you execute it.

This program requires the dataset file as the argument. For example, I execute the program at the command line as:

```
>perl Donnelly.Patrick.lab1.pl unique_tracks.txt
```

The initial template gives code to loop through each line of the file and prints out the line. You probably will not want to keep this line. Remember you can always use Ctrl (or Cmd) +C to cancel execution of the program.

Pre-processing

Step 1: Extract song title

Each line contains a track id, song id, artist name, and the song title, such as:

```
TRWRJSX12903CD8446<SEP>SOBSFBU12AB018DBE1<SEP>Frank Sinatra<SEP>Everything Happens To Me
```

You are only concerned with the last field. As your first task, you will write a regular expression that extracts the song title and stores it as the variable `$title`. You will discard all other information.

Step 2: Eliminate superfluous text

The song title, however, is quite noisy, often containing additional information beyond the song title. Consider this example:

```
Strangers In The Night (Remastered Album Version) [The Frank Sinatra Collection]
```

You need to preform some pre-processing in order to clean up the song titles. You will write a series of regular expressions that match a block of text and replace it with nothing.

Begin by writing a regular expression that matches a left parenthesis and any text that follows it. You need not match the right parenthesis explicitly. Replace the parenthesis and all text that follow it with nothing.

Repeat this for patterns beginning with the left bracket, the left curly brace, and all the other characters listed below:

```
(  [  {  \  /  _  -  :  "  `  +  =  *  feat.
```

Note that the above lists the left quote (on the tilde key above tab) and not the apostrophe (located near the enter key). We do not want to omit the apostrophe as it allows contrac-

tions. The last one is the abbreviation for `feat.`– short for featuring and followed by artist information you do not need to retain.

These steps will catch and fix the vast majority of irregularities in the song titles.

***Step 3:* Eliminate punctuation**

Next, find and delete the following typical punctuation marks:

```
? ¿ ! ¡ . ; & $ @ %
```

Unlike before, delete only the symbol itself and leave the text the follows. Be sure to do a 'global' match in order to replace all instances of the punctuation mark. Be careful to match the period itself as the symbol . has special meaning in regular expressions.

***Step 4:* Set to lowercase**

Convert all words in the sentence to lowercase. Perl has a function to do this for you.

Bi-gram Counts

A bigram is a sequence of two adjacent words in a text. The frequency distribution of bigrams in text(s) is commonly used in statistical natural language processing. Across the corpus of one million song titles, you will count all the bigrams.

First, you should split the title string into individual words. Next, you should use one or more data structures to keep track of these word pair counts. That is, for every word, you must keep track of the count for each word that follows it. I recommend you design your data structure for fast retrieval.

Lab Questions

Use your data structure(s) to answer the following questions.

Question 1: Which word most often follows the word "the"?

Question 2: How many words (total) follow the word "love"?

Question 3: How many different (unique) words follow the word "love"?

Question 4: Which word most often follows the word "love"?

Question 5: How many times does this word follow "love"?

Building a Song Title

Now you are going to built a probabilistic song title. First begin by creating a function "most common word" `mcw(·)`. This function will take in one argument, some word, and returns the word that most often followed that word in the dataset. If you find a tie, use the Perl function `rand()` to break it (i.e., a coin toss). For example, the line `print mcw(‘‘love’’);` should give you your answer to Question 4.

Now you are going to use this function to string together a song title. Beginning with a given starting word, write an iterative structure that strings together words that most commonly follow each other in the dataset. Continue until a word does not have a successive word in the dataset, or the count of words in your title reaches 20.

User Control

Next write a loop that repeatedly queries the user for a starting word until they choose to quit. Your program will ask:

```
Enter a word [Enter 'q' to quit]:
```

For each word entered, use your code above to create a song title of 20 words (or less). Print out your newly designed song title. Repeat, querying the user for a new word.

Lab Questions

Question 6: Using the starting word "love", what song title do you get?

Question 7: Using the starting word "amor", what song title do you get?

Question 8: Using the starting word "hey", what song title do you get?

Question 9: Using the starting word "little", what song title do you get?

Question 10: Try a few other words. Which phrase do you most often find?

Stop Words

Next try to fix the aforementioned problems. In NLP, common stop words are words that are often filtered out, such as common function words and articles. Before taking your bigram counts, filter out common the following common stop words from the song title:

```
the, a, an, and, or, to, of, for, from
```

Lab Questions

Question 11: Using the starting word "little" again, what song title do you get now?

Question 12: Did this solve the problem from question 10? Explain.

Question 13: Using the starting word "love" again, what song title do you get now?

Question 14: Did this solve the problem from question 6? If yes, give (here) your updated answer to Question 6. If no, propose a way to fix this problem and explain in detail.

Question 15: Try several words. Find a song title that terminates in less than 20 words. Which song title did you find?

Troubleshooting

This lab requires an independent study of the Perl language. You are encourage to use any web tutorials and resources to learn Perl. Given the size of the class, I will not be able to debug your code for you. Please do not send panicked emails requesting I fix your bug for you. Allow yourself plenty of time, and use patience, perseverance, and the internet to debug your code.

Lab Questions

Question 16: Name something you like about Perl. Explain.

Question 17: Name something you dislike about Perl. Explain.

Question 18: Did you enjoy this lab? Which aspects did you like and/or dislike?

Question 19: Approximately how many hours did you spend on this lab?

Question 20: Do you think you will use Perl again? For which type(s) of project(s)?

Extra Credit (10 pts)

Implement the "fix" you describe in Question 14. For extra credit you should fix the problematic phenomenon you observed in Question 6. Once you have done this, you can remove the restriction of 20 words maximum in the song title.

Question EC: Using the starting word "love" yet once again, what song title do you get this time? Describe how your extension fixed the problem.

Submission

Each student will complete and submit this assignment individually. Do not consult with others. However, you are encouraged to use the internet to learn Perl.

Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

Save the final version of your program as `[lastname]_[firstname].lab1.pl`. If you completed the extra credit, submit that version. You may comment out the code segments you used to answer the lab questions.

Type your lab questions in plain text as `[lastname]_[firstname].lab1.txt`. Include your name in the text file.

Your program must operate on the original dataset. Do not submit the song file dataset. I will use the file `unique_tracks.txt` from the source given above to evaluate your program.

You will email these files to `msucsci305@gmail.com`. Use the subject line `[lastname]_[firstname] Lab1 Submission`. Do not archive your files but instead use two attachments. Email your two files before 11:59pm on the due date. Late submissions will not be accepted.