CSCI 658 Fall 2023

HOMEWORK #1

Generation via n-gramsMERGEFORMATMERGEFORMAT

Purpose

1. Ensure that everyone can use Python on Jupyter (on any platform of your choice), including items such as dictionaries, list comprehensions, command line arguments, sorting, formatting, basic I/O and file encodings.

2. Evaluate the performance of generation via n‑grams, with and without smoothing, for different values of n.

3. Create a baseline for more sophisticated types of generation.

Input:

1. Use the UTF‑8 version of The Adventures of Tom Sawyer provided on Project Gutenberg at https://www.gutenberg.org/ebooks/74, i.e., ./pg74.txt. I will also save this version in the unlikely event that this file is updated during the pendency of the homework.

You can tell that this file is UTF‑8 by reading the first few lines of the file. You could also make a guess (many programs do) by looking at the frequencies of bit patterns. However, character files on Unix do not contain metadata to provide an unambiguous identification of encodings, so they need to be labeled.

Sample output:

I will post sample output over the weekend. Your output should match mine.

Submission:

I will post submission instructions over the weekend.

# Data cleanup

- A sentence is a string that ends with ‘.’ or ‘?’ or ‘:’. Add sentence markers <s> and </s> at the beginning and end of each sentence.

- Replace every em-dash by a space.

- Convert everything to lower case.

- Normal python split-on-whitespace to make tokens

- Leave ‘-’ and “’” when they are surrounded by letters, but drop them otherwise. Drop all other special characters.

# Text generation

1. Do random sentence generation with the raw n-gram counts. For example, suppose the word peter occurred 15 times, and suppose that the bigram list contained

- ‘Peter Pan’ with a count of 8

- ‘Peter Cottontail’ with a count of 2

- ‘peter out’ with a count of 5

Then to generate a word following ‘peter’, pick a random number from 1 to 15 and compare it to the cumulative totals 8 (for pan), 10 (for cottontail), and 15 (for out).

Generation starts with <s> and ends when </s> is randomly generated.

2. Do unigram, bigram, etc. generation up to n = 6.

3. Repeat step 2 using add‑1 smoothing.

4. Write up what you learned from this exercise, i.e., write a couple of paragraphs to compare the results of different values of n, and of the relevance of smoothing.

# Rules

You can use data structures and any standard software in the Python ecosystem, e.g., Pandas, numpy, etc.

# Coding suggestions

I used straight Python for this assignment.

I used defaultdict(int) as the basic data structure so that missing entries would automatically come up as zero.

I implemented the unigram counts as a dictionary with a count for each token.

I implemented the bigram counts as a dictionary of dictionaries. The inner dictionary has entries only for second words that occur, i.e., bigrams that don’t won’t have entries. The dictionary contains a raw count.

I implemented trigrams the same way, i.e., three levels of dictionary with non-occurring combinations not represented.

# Grading:

12 outputs @ 5 points each, plus a writeup and summary totals. There will be a penalty for violating any of the rules in the syllabus (e.g. late homework), academic integrity policy, course administration guide or programming style guide.