CS179G: Senior Project in Big Data Analytics

UC Riverside

Prof. Mariam Salloum

Problem Set 1

This problem set is due Wednesday, October 26th at 11:59pm. You encouraged to discuss the assignment with your classmates, but eventually each student should sit-down and work on their own code.

1 Introduction

An N-gram is a contiguous sequence of N words from a given sequence of text. An N-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n-1). N-gram models are widely used in statistical natural language processing. In speech recognition, phonemes and sequence of phonemes are modeled using a N-gram distribution. For sequences of words, the 1-grams (aka unigram) generated from 'We analyze large dataset' are ('We', 'analyze', 'large', 'dataset'). For the same sentence, the 2-grams (aka bigram) are ('We', 'We analyze', 'analyze large', 'large, dataset', 'dataset').

N-grams are used for various applications such as approximate matching, plagiarism detection,searching for the similar documents, automatic authorship detection, and linguistic cultural trend analysis. Google's Ngram Viewer is a good example of N-gram analysis.

In this assignment, you will create N-gram profile of the corpus of modern English literature. In this assignment, you will (1) extract all the bigrams, (2) compute the frequency of each bigram, and (3) rank the bigrams based on these frequencies.

As an example corpus for this assignment, you will be provided a dataset from the Gutenberg project. You will also be given a smaller dataset for testing purposes. Your will be required to :

- Use MapReduce to compute the frequency of bigrams written in Java (Note, this is similar to Lab 1)
- Use Spark to compute the frequency of bigrams written in Java or Python (Note, this is similar to Lab 2)
- Test your code on the cluster

2 Programming Requirements

You are required to use Java or Python for this assignment.

3 Generating N-gram profile

In this assignment, you should generate bigram profiles. To extract unigram, you should tokenize the sentences based on the whitespace characters. For the words that span sentence boundaries should be omitted. Add sentence beginning ('_START_') and end tokens ('_END_') for bigrams. This allows you to distinguish Ngrams that appear in sentence-medial positions from unigrams that occur at sentence boundaries later on. Ignore tense, gender, and hyphened words. 'He' and 'She' should be considered as the different words. 'have' and 'has', or 'have' and 'had' should be considered as the different words. 'well-described' should be considered as 1 unigram. Please ignore lower, upper cases. Convert all of the upper cases to lower case for your convenience. Your Ngrams should ignore any punctuations or apostrophes. Consider youre as a unigram. In your output, 'youŕe' should be listed as 'youre'. The frequency of Ngram should count only the exact same appearance of text. Do not eliminate the stop words.

Your corpus will be a set of modern English literature. Your Ngram profile should generate,

- Part A: A list of bigrams with their frequency using MapReduce
- Part B: A list of bigrams with their frequency using Spark

4 Input data

Your input data is a set of ebooks that are from Project Gutenberg. Use the small dataset given for Lab 1. Since we are computing bigrams which can blowup in size, we want to keep the dataset small to avoid a long run-time.

5 Submission

You will be submitting your code (as a ZIP file) via ILearn. Your submission should include a README file, all your source files, and the output data (for the provided dataset /dots not the test dataset). The README file should include a short description of what is included in the submission. In addition to submitting your files via iLearn you will demo your program to the TA during lab.