Programming Assignment 2

Content Based Authorship Detection using TF/IDF Scores and Cosine Similarity

Due: March 23nd, 2017 By 5:00PM
Submission: via Canvas, individual submission

Objectives

The goal of this programming assignment is to enable you to gain experience in:

- Creating an authorship identification system based on the similarity of the author's attributes
- Calculating TF/IDF scores using MapReduce
- Calculating Cosine Distance using MapReduce

1 Introduction

The goal of this assignment is to build an authorship detection system that provides a ranked list of possible authors for a document whose authorship is unknown. The system uses word uni-grams that have been used in programming assignment 1.

Finding the *attributes of authorship* is important to detect unique writing styles for each author. We will build an attribute vector for each of the authors and determine the authorship based on the similarity between pre-computed attribute vectors. We will use TF-IDF (Term Frequency Inverse Document Frequency) to calculate the weights of each of the entities in the attribute vector.

1.1 Term Frequency

Suppose we have a collection of documents written by M authors. This collection of documents may contain multiple books written by an author. Let's define a set of documents written by same author as a sub-collection j. We define f_{ij} to be the frequency (Number of occurrences) of term (word) i in sub-collection j.

$$TF_{ij}=0.5+0.5(f_{ij}/max_k f_{kj})$$

In this assignment, we use the augmented TF to prevent a bias towards longer documents. E.g. raw frequency divided by the maximum raw frequency of any term k in the sub-collection j. During this process, you should not eliminate stop words. The most frequent term in the sub-collection will have a

augmented TF value of 1.

1.2 Inverted Document Frequency

Suppose that term i appears in n_i sub-collections within the corpus. For this assignment, we define the IDF_I, as:

$$IDF_i = log_{10}(N/n_i)$$

where, N is the total number of sub-collections (number of authors).

1.3 TF.IDF value

The TF-IDF score is defined as TF_{ij} x IDF_i. The terms with the highest TF-IDF score are considered the best words that characterize the document.

1.4 Calculating Cosine Distance

The result of performing calculations outlined in sections 1.1 through 1.3 is that each set of books written by the same authror in your corpus will have an *Author Attribute Vector (AAV)*,

$$AAV_m = (TF.IDF_{word1}, TF.IDF_{word2}, TF.IDF_{word3}, TF.IDF_{word4}, \ldots..., TF.IDF_m)$$

Every author will have his or her own AAV representing writing style. To compare between an arbitrary AAV (from the document with unknown authorship) and existing $AAVs$, all of the $AAVs$ must have the same dimension. The $AAVs$ will be used to calculate the Cosine Distance to measure the similarity between the authors' writing styles. Suppose that we have two authors with vectors, $AAV_1 = [x_1, x_2, \ldots x_m]$ and $AAV_2 = [y_1, y_2, \ldots, y_m]$. The Cosine Similarity between them is defined as,

$$CosSimilarity = \cos(\theta) = \frac{A \bullet B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Finally, your system should be able to provide a ranked list of authors (top 10 most similar authors) for a document with unknown authorship.

To parse this document, please follow the same requirements that you followed for N-gram analysis (in Assignment 1).

You should tokenize the sentences based on whitespace characters. Ignore tense, gender, and hyphenated words. "He" and "She" should be considered as different words. "have" and "has", or "have" and "had" should be considered as different words. "well-described" should be considered as a word. Please don't distinguish between lower or upper cases of a word.

Your software should ignore any punctuations or apostrophes. Consider "you're" as a word. In your output, "you're" should be listed as "youre". The frequency of the term should count only the exact same appearances of the text. Do not eliminate the stop words (e.g. "the", "a", "is", "am").

2. Software Requirements

Your software should perform two sets of functionalities: off-line computing and command-line software.

2.1 Off-Line computing

This phase includes multiple steps;

(1) You should calculate the TF, IDF, and $TF\text{-}IDF$ values for all terms for all of the sub-collections in your corpus. **You are required to use MapReduce(s) for this step**. Custom implementations without using MapReduce is disallowed.
(2) You should create the $AAVs$ (author attribute vectors) for every author.
(3) You should store the results (author attribute vectors) in a HDFS file.

2.2 Command-Line software

(1) Read a document with unknown authorship and create an attribute vector for it. Make sure the dimensionality of this vector should be identical to the one used in the off-line computing (section 2.1).
(2) You should calculate the Cosine Similarity between the author attribute vector for this document and all of the author attribute vectors calculated in the section 2.1, and select top 10 authors. **You are required to use MapReduce for this step. Also, you are required to use a Combiner for this step.**

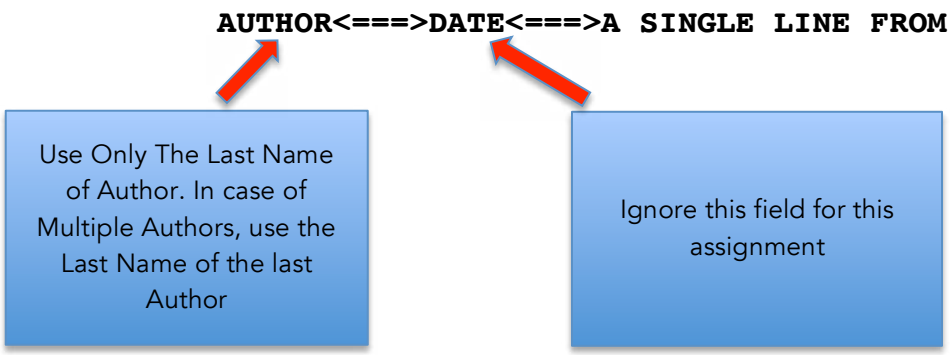
3 Input data

You will be working on the same 1gb input file that used for PA1, compiled from books from Project Gutenberg.

Click here to download this file.

You are not required to submit the output of your software on running on this 1gb file. Keep your intermediate outputs stored in your HDFS for use later on in the offline version of your software.

Each line in this file will look like the example below:



Example:

Arthur Conan Doyle<===>June 19, 2008<===>"Well, by his insufferable rudeness and impossible behavior."

You will also find a smaller datafile for testing out your software on this link. This file would be about 7mb in size and is to be used for testing purposes only. You should not run your offline software based on outputs received from this input file. This is for testing purposes only.

4. Submission

This assignment must be done **individually**. Please submit the tar ball of your source files along with any jar you need to run your software via Canvas. The source files should include your java code for Mapper/Reducer functions and any script file that you will use for the demo. You will download your jars from Canvas for the demo. Do not miss any files. During your demo, you are not allowed to use any file outside your Canvas submission.

5. Grading

Each of the submissions will be graded based on the demonstration of your software. During the demonstration, you should present:

Step 1. Off-line computing with the test dataset [5 points]

Step 2. Execute your command line software. [5 points]

Demos include short interviews about your software design and implementation details. Each question and answer will count toward your score. **This assignment will account for 10% of your final course grade.**

6. Late policy

Please check the late policy posted on the course web page.