Lab Course: Distributed Data Analytics
Exercise Sheet 6

Mofassir ul Islam Arif
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: Monday 10:00AM (on LearnWeb, course code: 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit two items. a) a zip or a tar file containing python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in the form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.

In this exercise sheet, you will extend the simple WordCount program shipped with Hadoop. Instead, you are going to implement your own `mapper.py` and `reducer.py` and apply to Hadoop Streaming jobs. A pipeline of submitting Hadoop Streaming jobs is already introduced in the lab course's lecture notes.

In this exercise sheet, you are going to apply MapReduce techniques to pre-process text data in natural language processing (NLP). NLP is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human languages. In big data scenario, NLP is concerned with programming computers to fruitfully process large natural language corpora.

More precisely, you are going to do some basic tasks in NLP including data cleaning, text tokenization and convert words into their Term Frequency, Inverse Document Frequency (TFIDF) scores.

Exercise 1: Basic Map and Reduce (2 Points)

In this exercise, you are going to build a basic version of WordCount program by implementing `mapper.py` and `reduce.py` files. The expected result in this exercise is similar to the result that you get from the shipped WordCount program. Thus, you can run the shipped WordCount to check the behaviors of your implementation. The text dataset used for this exercise can be found here: `http://www.gutenberg.org/files/2591/2591-0.txt`.

More precisely, you are going to answer the following questions for completing this exercise:

1. What are your `mapper.py` and `reduce.py` solutions?
2. Describe step-by-step how you apply them and the outputs during this procedure.

Exercise 2: Data cleaning and text tokenization (6 points)

Data cleaning is absolutely crucial for generating useful datasets. By adding some functions to your MapReduce solution, you are going to clean raw data by applying some steps belows:

1. Cleaning: remove all punctuations and numbers.
2. Stopping: removing meaningless words. The list of common English stopwords used in this exercise sheet can be found in the reference [4].

Several raw datasets used for exercise 2 and exercise 3 are follows:

1. Raw text 1: `http://www.gutenberg.org/files/2591/2591-0.txt`

2. Raw text 2: `http://www.gutenberg.org/files/1400/1400-0.txt`
3. Raw text 3: `http://www.gutenberg.org/files/219/219-0.txt`
4. Raw text 4: `http://www.gutenberg.org/files/4300/4300-0.txt`
5. Raw text 5: `http://www.gutenberg.org/files/158/158-0.txt`

In order to complete this exercise, you are asked to describe step-by-step how you clean and tokenize the raw data.

Exercise 3: Calculate TFIDF scores of words/tokens (12 points)

Before you do this exercise, let's read some tutorials about TFIDF and how to calculate the scores. You can start with references [2,3] in the Annex section or you can search your own tutorials. The summarization of how to calculate TFIDF scores can also be found in the TFIDF section.

In order to complete this exercise, you are asked to describe step-by-step how you get the final data.

TFIDF

Typically, the TFIDF score is contained by two components: the first component computes the normalized Term Frequency (IF) which means the number of times a word appears in a document, divided by the total number of words in that document; the second component is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific token appears.

TFIDF score is expressed as

$$TFIDF = TF * IDF \,, \tag{1}$$

where TF represents the importance of a token in a document, and IDF represents the importance of a token in the entire text collection.

A TF could simply be represented as the number of times a token occurs in the document, but it is more commonly that you normalize it by taking into account the total number of tokens appear in the document, so that the overall score account for document's length relative to a token's frequency. Thus, the TF is often divided by the document's length as a way of normalization:

$$\mathrm{TF}(t, d) = \frac{n^d(t)}{|d|} \,, \tag{2}$$

where $n^d(t)$ is the number of times a token t appears in a document d and $|d|$ is the total number of tokens in the document d.

An IDF could simply be represented as the logarithm of a quotient that is defined by the number of documents in the entire corpus divided by the number of documents in the corpus that contains the token. While the TF is calculated on a per-document basis, the IDF is computed on the basis of the entire corpus. Thus, the IDF is calculated as follows.

$$\mathrm{IDF}(t) = \log \frac{|C|}{n^C(t)} \,, \tag{3}$$

where $|C|$ is the total number of documents in the corpus and $n^C(t)$ is the number of documents that contains token t.

The expected outcome of this exercise, as well as the whole exercise sheet, is the final data that contain tokens and their TFIDF scores.

Annex

Some good references that you might need to learn before doing each exercise can be found here:

1. Hadoop Linux Setup urlhttps://phoenixnap.com/kb/install-hadoop-ubuntu
2. Introduction to Natural Language Processing (NLP):
   `http://blog.algorithmia.com/introduction-natural-language-processing-nlp/`
3. TFIDF `http://www.tfidf.com/` and `https://en.wikipedia.org/wiki/Tf%E2%80%93idf`
4. Tutorial: Finding Important Words in Text Using TF-IDF
   `http://stevenloria.com/finding-important-words-in-a-document-using-tf-idf/`
5. Common English stopwords: `http://www.textfixer.com/tutorials/common-english-words.txt`
6. Data-Intensive Text Processing with MapReduce
   `http://www.umiacs.umd.edu/~jimmylin/MapReduce-book-final.pdf`