INF344, Télécom ParisTech

Inverted Index with MapReduce

Pierre Senellart (pierre.senellart@telecom-paristech.fr)

18 May 2016

The purpose of this lab session is to build an inverted index out of a collection of text documents using the MapReduce programming model.

1 Baseline

A skeleton Eclipse project is available for download from the course Web site. You are free to use another Java development environment; if you do not use Eclipse, you will need to make sure both the `config/` directory and all JARs in the `lib/` directory are in the class path.

The skeleton project contains the following JAVA source files:

InvertedIndex.java This is the main part of the program to implement and the only file you should modify. When submitting your solution to the submission server, you only need to submit this file. *TODO* annotations indicate which parts of the file need to be filled in.

PairTextDoubleWritable.java A Hadoop Writable class (i.e., a type suitable for reading and storing within the MapReduce framework) to store both a string (as a Hadoop Text object) and a floating-point number (as a Hadoop DoubleWritable object). You can access both components with the getFirst() and getSecond() methods.

PairWritable.java An abstract superclass of PairTextDoubleWritable.

Stemmer.java An implementation of Porter's stemming algorithm. To use it, first create a Stemmer object using the default constructor, and then use the method stem(String) on every string to stem.

PublicTests.java A collection of public tests that will be run on your implementation.

BaseTests.java A superclass of PublicTests with some utility methods.

The `data/` directory contains two corpora (extracted from the Simple English Wikipedia at `http://simple.wikipedia.org/`) and a list of stop words. We recommend using the `mini.txt` corpus for testing purposes, as is done in the PublicTests class. Public tests are configured to produce output within `data/resultDirectory/`. Do not change the content of the `config/` and `lib/` directories. If you need to debug a specific part of your code, do not hesitate creating a very small corpus and examine the output of your code on this small corpus.

2 Submission and Evaluation

Submit your solution by Tuesday, May 24, 11:59pm on the submission server for full credit. Submissions received after this date and before Wednesday, May 25, 1:30pm will incur a penalty of 4 points out of 20. Submissions received after this second deadline will not be considered. As previously mentioned, for this lab you only need to submit the `InvertedIndex.java` file, not the whole Eclipse project.

Your submission will first be run against the public tests; as they are identical to the ones provided in the PublicTests class, there is no need to use the submission server unless your submission passes these tests locally. A score out of 20 points will be given as an indication (this score is not part of the grade). When confident about your submission, you can release it to test it against the secret (release) tests. The score obtained on the release tests will be used as a grade for this lab session. You are allowed at most 3 release submissions per hour. Evaluation of your submissions will typically take several minutes.

3 Structure of `InvertedIndex.java` and What to Implement

alphaDigit is a constant regular expression pattern used for tokenization; you do not need to use it directly.

corpusSize is a shared counter for the number of documents in the corpus. You should increment it where relevant.

getCorpusSize() returns the number of documents currently indexed. You should implement this function.

map(Text key, Text value, Context context) is the Map function to implement. It receives as input the document title (`key`) and the document content (`value`). It should output for each term occurrence (with `context.write(Text,PairTextDoubleWritable)`) the term (as key) and a pair (as value) with the document title and the tf weight. The tokenization loop is already provided. Tokens should be stemmed and lowercased. Stop words should be discarded. When computing tf, consider that the size of the document is the number of terms after stop word removal. For evaluation purposes, for a given document, terms should be output in *lexicographic order* by the Map function.

reduce(Text key, Iterable<PairTextDoubleWritable> values, Context context) is the Reduce function to implement. It receives as input a term (`key`) and an iterable list of pairs with document titles and tf weights. This function should compute each posting list of the inverted index in the following form:

```
documentTitle1:tfidf1\tdocumentTitle2:tfidf2\t...
```

where `documentTitle1` is the title of the document with the highest tf-idf score for this term, and `tfidf1` the corresponding tf-idf score, rounded to 5 decimals after the decimal point. title/score pairs are separated by a tabulation character. The output of the Reduce function (with `context.write(Text,Text)`) is the term (as key) and the posting list sorted by decreasing order of tf-idf scores (as value). When computing idf, use the logarithm to the base 2 ($\log_2$). *Hint: Use a container of `DocumentWeight` objects for in-memory storage and ordering of the posting list.*

retrieveStopWords(String filename) should load the stop word list referenced in the `filename` string (with one stop word per line in the file) and store them in an appropriate in-memory data structure. Stop words should be stemmed as they will be compared to stemmed tokens. You should implement this function.

isStopWord(String token) returns whether a word is a stop word. You should implement this function.

buildInvertedIndex(String index, String output) is the function that launches the MapReduce job. It is already implemented, no need to change it.

DocumentWeight is an inner static class for in-memory storage of document title and weights. You should implement a comparator for this function that will allow sorting of the posting lists. Be careful to implement an ordering *consistent with equality* (see the discussion in `http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html`).