Lab 1.3 Report

Hendrik Molder (9985362)
COMP38120

February 15, 2019

Contents

Introduction

This report describes the author's approach – including a brief discussion of the functionality and performance – of the `BasixInvertedIndex.java` program. An example of the index and a listing of the program are attached to this report (please see Appendices 1 and 2 respectively).

1 Implementation Notes

1.1 Implementation of the Mapper

The mapper produces the output in the following format

```
Text, PairOfWritables<
    TripleOfIntsString,
    ArrayListWritable<IntWritable>
>
```

As seen above, for each *unique* term the mapper emits the token followed by a `PairOfWritables`, which consists of two parts – a `TripleOfIntsString` for representing (1) the count of the token appearances in the document, (2) the total count of the tokens in the document and (3) the filename, and a `ArrayListWritable` for storing the positions of the token in the document.

1.2 Implementation of the Reducer

The index (i.e. the output of the reducer) is structured as follows

```
Text, ArrayListWritable<
    PairOfWritables<
        PairOfWritables<Text, String2FloatOpenHashMapWritable>,
    ArrayListWritable<IntWritable>>>
>
```

An example of the index for a token `banana` is provided below. The token is followed by an `ArrayList` containing entries for each document associated with the token. The entries consist of two parts – a metadata `PairOfWritables` and an `ArrayList` of the positions of the token in the

document – stored in a `PairOfWritables`. The metadata `PairOfWritables` has two objects – the document name (file name) and a `String2FloatOpenHashMapWritable` for storing useful numerical metadata, e.g. TF, IDF, and $TF - IDF$ scores.

```
banana [((Bart_the_Genius.txt.gz,
tfidf=>7.793647E-4, idf=>1.7917595, tf=>4.3497173E-4), [850])]
```

The downside of this structure is that metadata that is not document dependent is duplicated – for example, the IDF score that is the same for a token in a collection, is added to the metadata of each entry. At the same time, the TF score is relevant to each document (although it is not a good measure to be used alone to evaluate the relevance of the term in a document).

2 The Functionality of the Index

2.1 Tokenisation

The `StringTokenizer` was used for tokenisation (see line 129 in Appendix 2). The author decided to break the input at the following characters and symbols: `[space]'",;:.()[]{}!?/#$&^%£|=`. Some symbols like hyphens (`-`) were not included in the tokeniser to keep the original form of the dates and other terms (e.g. `2012-06-21`). However, this approach could mean that some terms – for example, `pro-Arab`, `Manchester-based` and others – that should be two tokens will be indexed as one.

2.2 Stemming

The standard stemming function was used in the inverted index. Stemming (i.e. removing suffixes) allows us to collapse similar forms to a canonical form. For example, tokens `belief` and `believing` could be indexed as `belie`. The problem that is introduced along with the benefits of stemming, is over-stemming – as a result, the index contains tokens like `d`, `dai`, `en` and others, which many conflate unrelated forms.

2.3 Case Folding

The case folding function is implemented on line 105 of Appendix 2. A fairly basic approach was used for case folding – the author decided that acronyms (e.g. DVD, PHP, VHS, BBC) should not be converted to lower-case. The implementation leaves all tokens in their original form, if they contain **more than 1** upper-case characters. This means that terms like `JavaScript` and `X-Files` also keep their original forms.

This approach offers a small performance improvement as the case folding function can return after seeing the second upper-case character as compared to checking if every single character is upper-case (the approach we would have used in case we wanted to look for acronyms only).

2.4 Positional Indexing

The index provides quite a lot of information about the documents. For example, a list of positions of the token is provided for each documents. In theory, it could support queries of phrases – e.g. for a query `'bart or bart'` you could see if any document has the term `bart` in positions x and $x + 2$, where x is the first occurrence of `bart`.

The fact that documents could be bigger than a single file split (i.e. document is split into multiple parts for the mapper to process) was not taken into account – this is a potential weakness of the approach.

2.5 Document and Term Frequency

In addition to Positional Indexing, TF, IDF, and $TF - IDF$ (see the formulae used below) scores are provided for each document associated with the token. $TF - IDF$ shows the relative frequency of a token in a document compared to the inverse of the proportion of that token over the collection of documents. Hence, tokens that are frequent in small number of documents have higher $TF - IDF$. An example use case of the $TF - IDF$ score would be the ordering of the results starting from the (potentially) most relevant document.

However, the pay-off of calculating the $TF - IDF$ scores is efficiency – an extra loop over the token occurrences was required in the reducer (see line 228 of Appendix 2). Ramos (2003) also points out that although $TF - DF$

is a good benchmark, it cannot understand the relationship between words – this is why $TF-IDF$ is almost always combined with some other technique.

$$TF = \frac{\texttt{number of times a token appeared in the document}}{\texttt{total number of tokens in the document}}$$

$$IDF = \log_e(\frac{\texttt{total number of documents}}{\texttt{number of documents with the token in it}})$$

$$TF - IDF = TF * IDF$$

2.6 Flagging of Important Terms

The flagging of important terms was not implemented. However, as for each term the index contains a filename and a $TF-IDF$ score, this could easily be added to the index.

3 Performance and Efficiency Analysis

3.1 In-Mapper Aggregation

By using local aggregation in the mapper (see line 162 in Appendix 2), we were able to reduce the amount of intermediate data generated. Instead of emitting a key-value pair for each term in a document, we collate the terms into a hash map and later emit each *unique* term – this approach is often referred to as *in-mapper combining* (Lin J. 2010).

3.2 Stop Word Removal

Stop word removal was also used, as required - this offers a significant improvement in terms of efficiency. As a result of removing words like `the` and `a`, we save a lot of memory space and time as we do not have to count and store all occurrences of these words.

3.3 Possible Improvements

Obviously, quite a few improvements can still be made. I have listed some of the most obvious ones below.

1. **Structure of the index** – instead of storing metadata in a hashmap for each document, the scores that are the same for each document (e.g. $TF - IDF$ and IDF, could be stored in a different way. This would have an affect on the memory we use as it would decrease the amount of duplication.

2. **Tokenisation** – as mentioned the '`-`' symbol (and in fact other symbol, such as '`+`') were not included in the `StringTokenizer`. Although it helps to preserve some terms that should definitely be a single token (e.g dates in the form `06-11-2016`, which would lose meaning if we were to tokenise these separately as `06`, `11` and `2016`), it also means that some terms that should be multiple tokens are now indexed as one.

3. **Considering synonyms** – Ramos (2003) suggests, that it might be a good idea to combine the $TF - IDF$ score with synonyms to improve the search quality. He illustrates his point with the following example: say a term `priest` has an extremely high $TF - IDF$ score; if an user searches for term `reverend`, we should display documents containing `priest`, but as $TF-IDF$ does not understand the relationship between words, this is not the case.

4. **Handling document splitting** – in this implementation, the document splits are not handled in the context of positional indexing. This is something that could be considered.

References

Lin J., Dyer C. (2010). *Data-Intensive Text Processing with MapReduce*. Morgan Claypool Publishers. URL: `http://lintool.github.io/MapReduceAlgorithms/`.

Ramos, Juan et al (2003). "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242, pp. 133–142.

Appendix 1. Index Output

```
1  2007-05-25 [((Bart_the_Murderer.txt.gz, {tfidf=>7.673488E-4,
       idf=>1.7917595, tf=>4.282655E-4}), [1871])]
2  2007-06-09 [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4,
       idf=>1.7917595, tf=>4.199916E-4}), [2056])]
3  2007-07-24 [((Bart_the_General.txt.gz, {tfidf=>9.807113E-4,
       idf=>1.7917595, tf=>5.473454E-4}), [1425])]
4  2007-07-26 [((Bart_the_Lover.txt.gz, {tfidf=>6.8257505E-4,
       idf=>1.7917595, tf=>3.8095238E-4}), [2409])]
5  2007-08-05 [((Bart_the_Genius.txt.gz, {tfidf=>7.793647E-4,
       idf=>1.7917595, tf=>4.3497173E-4}), [1772])]
6  2007-08-07 [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4,
       idf=>1.7917595, tf=>4.199916E-4}), [2134])]
7  2007-08-10 [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4,
       idf=>1.7917595, tf=>4.199916E-4}), [2079])]
8  2007-08-17 [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4,
       idf=>1.7917595, tf=>4.199916E-4}), [2154])]
9  DVD [((Bart_the_Fink.txt.gz, {tfidf=>0.0, idf=>0.0, tf
       =>0.007019186}), [82, 1444, 1470, 1710, 1718, 1733, 1741,
       1752, 1760, 1771, 1779, 1790, 1798, 1834, 1856]), ((
       Bart_the_Lover.txt.gz, {tfidf=>0.0, idf=>0.0, tf
       =>0.0038095238}), [58, 1744, 1867, 2149, 2175, 2194, 2217,
        2242, 2325, 2360]), ((Bart_the_Genius.txt.gz, {tfidf
       =>0.0, idf=>0.0, tf=>0.0069595478}), [78, 1424, 1477,
       1681, 1698, 1783, 1791, 1804, 1812, 1830, 1838, 1868,
       1876, 1894, 1902, 1956]), ((Bart_the_General.txt.gz, {
       tfidf=>0.0, idf=>0.0, tf=>0.0060207993}), [44, 940, 975,
       1277, 1288, 1304, 1372, 1380, 1396, 1404, 1489]), ((
       Bart_the_Murderer.txt.gz, {tfidf=>0.0, idf=>0.0, tf
       =>0.005567452}), [85, 1625, 1668, 1824, 1832, 1910, 1918,
       1965, 1973, 2004, 2012, 2078, 2102]), ((Bart_the_Mother.
       txt.gz, {tfidf=>0.0, idf=>0.0, tf=>0.0046199076}), [84,
       1783, 1886, 1967, 1975, 1990, 1998, 2011, 2019, 2119,
       2163])]
10 Dum-Doodili [((Bart_the_Fink.txt.gz, {tfidf=>8.3844614E-4,
       idf=>1.7917595, tf=>4.679457E-4}), [116])]
11 EARLY [((Bart_the_General.txt.gz, {tfidf=>9.807113E-4, idf
       =>1.7917595, tf=>5.473454E-4}), [1456])]
12 EMCSQU [((Bart_the_Genius.txt.gz, {tfidf=>7.793647E-4, idf
       =>1.7917595, tf=>4.3497173E-4}), [1098])]
13 ESPN [((Bart_the_Lover.txt.gz, {tfidf=>0.0013651501, idf
       =>1.7917595, tf=>7.6190475E-4}), [2094, 2444])]
14 GRABS [((Bart_the_General.txt.gz, {tfidf=>9.807113E-4, idf
       =>1.7917595, tf=>5.473454E-4}), [1455])]
```

```
15 GoodFella [((Bart_the_Murderer.txt.gz, {tfidf=>0.0030693952,
      idf=>1.7917595, tf=>0.001713062}), [819, 870, 1210, 1789])
      ]
16 Homediddly-Dum-Doodili [((Bart_the_Fink.txt.gz, {tfidf
      =>8.3844614E-4, idf=>1.7917595, tf=>4.679457E-4}), [2019])
      ]
17 URL [((Bart_the_General.txt.gz, {tfidf=>9.807113E-4, idf
      =>1.7917595, tf=>5.473454E-4}), [1825])]
18 US [((Bart_the_General.txt.gz, {tfidf=>0.0012026407, idf
      =>1.0986123, tf=>0.0010946908}), [1204, 1228]), ((
      Bart_the_Genius.txt.gz, {tfidf=>4.7786528E-4, idf
      =>1.0986123, tf=>4.3497173E-4}), [1669])]
19 VHS [((Bart_the_Genius.txt.gz, {tfidf=>0.0014335959, idf
      =>1.0986123, tf=>0.0013049152}), [1620, 1650, 2003]), ((
      Bart_the_General.txt.gz, {tfidf=>0.0024052814, idf
      =>1.0986123, tf=>0.0021893815}), [1178, 1207, 1258, 1617])
      ]
20 about [((Bart_the_Fink.txt.gz, {tfidf=>0.0041922308, idf
      =>1.7917595, tf=>0.0023397286}), [183, 201, 1278, 2086,
      2104])]
21 academ [((Bart_the_General.txt.gz, {tfidf=>6.0132035E-4, idf
      =>1.0986123, tf=>5.473454E-4}), [1130]), ((Bart_the_Genius
      .txt.gz, {tfidf=>9.5573056E-4, idf=>1.0986123, tf
      =>8.6994347E-4}), [415, 590])]
22 accept [((Bart_the_Genius.txt.gz, {tfidf=>4.7786528E-4, idf
      =>1.0986123, tf=>4.3497173E-4}), [321]), ((Bart_the_Lover.
      txt.gz, {tfidf=>4.1851896E-4, idf=>1.0986123, tf
      =>3.8095238E-4}), [1432])]
23 accident [((Bart_the_Fink.txt.gz, {tfidf=>3.2435523E-4, idf
      =>0.6931472, tf=>4.679457E-4}), [278]), ((
      Bart_the_Murderer.txt.gz, {tfidf=>2.9685104E-4, idf
      =>0.6931472, tf=>4.282655E-4}), [317]), ((Bart_the_Mother.
      txt.gz, {tfidf=>8.73348E-4, idf=>0.6931472, tf
      =>0.0012599748}), [364, 597, 1195])]
24 accord [((Bart_the_Lover.txt.gz, {tfidf=>6.8257505E-4, idf
      =>1.7917595, tf=>3.8095238E-4}), [1121])]
25 account [((Bart_the_Fink.txt.gz, {tfidf=>0.0025153384, idf
      =>1.7917595, tf=>0.0014038371}), [481, 494, 1332])]
26 bar [((Bart_the_Murderer.txt.gz, {tfidf=>0.003836744, idf
      =>1.7917595, tf=>0.0021413276}), [250, 267, 461, 617,
      1201])]
27 bart [((Bart_the_Murderer.txt.gz, {tfidf=>-0.23104906, idf
      =>-0.6931472, tf=>0.33333334}), [1]), ((Bart_the_Murderer.
      txt.gz, {tfidf=>-0.016029956, idf=>-0.6931472, tf
      =>0.023126338}), [10, 83, 117, 147, 153, 172, 215, 245,
```

```
272, 286, 289, 303, 325, 435, 484, 504, 524, 569, 592,
630, 650, 668, 682, 747, 756, 772, 1140, 1147, 1157, 1193,
 1233, 1273, 1297, 1338, 1371, 1447, 1528, 1635, 1675,
1763, 1829, 1915, 1928, 1970, 2009, 2163, 2169, 2176,
2185, 2191, 2243, 2273, 2279, 2298]), ((Bart_the_Fink.txt.
gz, {tfidf=>-0.23104906, idf=>-0.6931472, tf=>0.33333334})
, [1]), ((Bart_the_Fink.txt.gz, {tfidf=>-0.011676789, idf
=>-0.6931472, tf=>0.016846046}), [10, 117, 158, 176, 240,
270, 313, 456, 487, 502, 529, 557, 577, 715, 793, 836,
1384, 1434, 1484, 1526, 1609, 1631, 1689, 1715, 1738,
1757, 1776, 1795, 1943, 1946, 1953, 1962, 1970, 2020,
2061, 2079]), ((Bart_the_Genius.txt.gz, {tfidf
=>-0.012964476, idf=>-0.6931472, tf=>0.018703785}), [10,
62, 103, 115, 178, 233, 279, 357, 378, 404, 422, 439, 461,
 484, 511, 536, 556, 596, 647, 739, 807, 952, 1078, 1144,
1218, 1260, 1371, 1404, 1451, 1462, 1522, 1767, 1835,
1873, 1899, 2117, 2145, 2151, 2158, 2167, 2175, 2220,
2232]), ((Bart_the_Genius.txt.gz, {tfidf=>-0.23104906, idf
=>-0.6931472, tf=>0.33333334}), [1]), ((Bart_the_General.
txt.gz, {tfidf=>-0.23104906, idf=>-0.6931472, tf
=>0.33333334}), [1]), ((Bart_the_General.txt.gz, {tfidf
=>-0.013658072, idf=>-0.6931472, tf=>0.019704433}), [10,
67, 79, 142, 169, 177, 230, 256, 274, 293, 298, 312, 326,
334, 351, 355, 387, 416, 488, 701, 797, 828, 914, 965,
1093, 1377, 1401, 1417, 1561, 1579, 1670, 1676, 1683,
1692, 1739, 1751]), ((Bart_the_Lover.txt.gz, {tfidf
=>-0.011618467, idf=>-0.6931472, tf=>0.016761905}), [10,
91, 121, 127, 146, 189, 222, 240, 339, 408, 555, 626, 773,
 793, 822, 861, 898, 970, 977, 1001, 1014, 1218, 1257,
1349, 1399, 1529, 1747, 1848, 1874, 1978, 2126, 2139,
2165, 2184, 2207, 2232, 2458, 2464, 2471, 2480, 2532,
2562, 2568, 2587]), ((Bart_the_Lover.txt.gz, {tfidf
=>-0.23104906, idf=>-0.6931472, tf=>0.33333334}), [1]), ((
Bart_the_Mother.txt.gz, {tfidf=>-0.23104906, idf
=>-0.6931472, tf=>0.33333334}), [1]), ((Bart_the_Mother.
txt.gz, {tfidf=>-0.017175844, idf=>-0.6931472, tf
=>0.024779504}), [10, 56, 117, 152, 240, 362, 396, 431,
458, 467, 501, 554, 564, 570, 588, 596, 605, 628, 631,
668, 689, 728, 733, 736, 810, 880, 938, 960, 1082, 1098,
1271, 1291, 1299, 1321, 1408, 1551, 1588, 1635, 1670,
1683, 1700, 1795, 1849, 1866, 1892, 1907, 1946, 1972,
1995, 2016, 2069, 2104, 2186, 2192, 2199, 2208, 2216,
2264, 2299])]
bob [((Bart_the_Fink.txt.gz, {tfidf=>0.0056550005, idf
=>1.0986123, tf=>0.005147403}), [39, 73, 133, 330, 339,
```

```
      704, 814, 845, 1100, 1543, 2036]), ((Bart_the_Lover.txt.gz
      , {tfidf=>4.1851896E-4, idf=>1.0986123, tf=>3.8095238E-4})
      , [1710])]
29 boi [((Bart_the_Mother.txt.gz, {tfidf=>4.614079E-4, idf
      =>1.0986123, tf=>4.199916E-4}), [731]), ((
      Bart_the_Murderer.txt.gz, {tfidf=>9.409955E-4, idf
      =>1.0986123, tf=>8.56531E-4}), [522, 1215])]
30 bolivian [((Bart_the_Mother.txt.gz, {tfidf=>0.0022575718, idf
      =>1.7917595, tf=>0.0012599748}), [758, 1445, 1483])]
31 bomb [((Bart_the_Fink.txt.gz, {tfidf=>5.140909E-4, idf
      =>1.0986123, tf=>4.679457E-4}), [1317]), ((
      Bart_the_General.txt.gz, {tfidf=>6.0132035E-4, idf
      =>1.0986123, tf=>5.473454E-4}), [369])]
32 book [((Bart_the_General.txt.gz, {tfidf=>0.0, idf=>0.0, tf
      =>5.473454E-4}), [884]), ((Bart_the_Mother.txt.gz, {tfidf
      =>0.0, idf=>0.0, tf=>4.199916E-4}), [1719]), ((
      Bart_the_Lover.txt.gz, {tfidf=>0.0, idf=>0.0, tf
      =>7.6190475E-4}), [777, 1920]), ((Bart_the_Genius.txt.gz,
      {tfidf=>0.0, idf=>0.0, tf=>0.0013049152}), [1181, 1321,
      1578]), ((Bart_the_Fink.txt.gz, {tfidf=>0.0, idf=>0.0, tf
      =>4.679457E-4}), [1503]), ((Bart_the_Murderer.txt.gz, {
      tfidf=>0.0, idf=>0.0, tf=>4.282655E-4}), [1540])]
33 execut [((Bart_the_Mother.txt.gz, {tfidf=>2.9111598E-4, idf
      =>0.6931472, tf=>4.199916E-4}), [996]), ((
      Bart_the_Murderer.txt.gz, {tfidf=>2.9685104E-4, idf
      =>0.6931472, tf=>4.282655E-4}), [605]), ((Bart_the_Lover.
      txt.gz, {tfidf=>2.6405606E-4, idf=>0.6931472, tf
      =>3.8095238E-4}), [1005])]
34 expect [((Bart_the_General.txt.gz, {tfidf=>9.807113E-4, idf
      =>1.7917595, tf=>5.473454E-4}), [263])]
35 expens [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4, idf
      =>1.7917595, tf=>4.199916E-4}), [1778])]
36 experi [((Bart_the_General.txt.gz, {tfidf=>7.587818E-4, idf
      =>0.6931472, tf=>0.0010946908}), [1000, 1107]), ((
      Bart_the_Lover.txt.gz, {tfidf=>5.281121E-4, idf
      =>0.6931472, tf=>7.6190475E-4}), [877, 1087]), ((
      Bart_the_Genius.txt.gz, {tfidf=>3.0149944E-4, idf
      =>0.6931472, tf=>4.3497173E-4}), [464])]
37 expertli [((Bart_the_Fink.txt.gz, {tfidf=>8.3844614E-4, idf
      =>1.7917595, tf=>4.679457E-4}), [1572])]
38 explain [((Bart_the_Mother.txt.gz, {tfidf=>4.614079E-4, idf
      =>1.0986123, tf=>4.199916E-4}), [754]), ((
      Bart_the_Murderer.txt.gz, {tfidf=>0.0014114934, idf
      =>1.0986123, tf=>0.0012847966}), [313, 679, 1566])]
39 explod [((Bart_the_Fink.txt.gz, {tfidf=>5.140909E-4, idf
```

```
    =>1.0986123, tf=>4.679457E-4}), [799]), ((Bart_the_Genius.
    txt.gz, {tfidf=>4.7786528E-4, idf=>1.0986123, tf
    =>4.3497173E-4}), [465])]
40  explor [((Bart_the_Mother.txt.gz, {tfidf=>4.614079E-4, idf
    =>1.0986123, tf=>4.199916E-4}), [1856]), ((Bart_the_Lover.
    txt.gz, {tfidf=>4.1851896E-4, idf=>1.0986123, tf
    =>3.8095238E-4}), [2436])]
41  explos [((Bart_the_Fink.txt.gz, {tfidf=>0.0010281818, idf
    =>1.0986123, tf=>9.358914E-4}), [788, 1210]), ((
    Bart_the_Murderer.txt.gz, {tfidf=>4.7049776E-4, idf
    =>1.0986123, tf=>4.282655E-4}), [44])]
42  expos [((Bart_the_Fink.txt.gz, {tfidf=>8.3844614E-4, idf
    =>1.7917595, tf=>4.679457E-4}), [279])]
43  full [((Bart_the_Genius.txt.gz, {tfidf=>4.7786528E-4, idf
    =>1.0986123, tf=>4.3497173E-4}), [903]), ((
    Bart_the_General.txt.gz, {tfidf=>0.0012026407, idf
    =>1.0986123, tf=>0.0010946908}), [754, 1065])]
44  full-length [((Bart_the_Mother.txt.gz, {tfidf=>0.0022575718,
    idf=>1.7917595, tf=>0.0012599748}), [278, 944, 966])]
45  fun [((Bart_the_Mother.txt.gz, {tfidf=>0.0013842238, idf
    =>1.0986123, tf=>0.0012599748}), [529, 1453, 1466]), ((
    Bart_the_Genius.txt.gz, {tfidf=>4.7786528E-4, idf
    =>1.0986123, tf=>4.3497173E-4}), [1449])]
46  funer [((Bart_the_Fink.txt.gz, {tfidf=>8.3844614E-4, idf
    =>1.7917595, tf=>4.679457E-4}), [1364])]
47  funni [((Bart_the_Fink.txt.gz, {tfidf=>1.8973566E-4, idf
    =>0.4054651, tf=>4.679457E-4}), [1010]), ((Bart_the_Mother
    .txt.gz, {tfidf=>1.7029194E-4, idf=>0.4054651, tf
    =>4.199916E-4}), [2080]), ((Bart_the_General.txt.gz, {
    tfidf=>2.2192945E-4, idf=>0.4054651, tf=>5.473454E-4}),
    [1540]), ((Bart_the_Murderer.txt.gz, {tfidf=>1.7364672E-4,
     idf=>0.4054651, tf=>4.282655E-4}), [922])]
48  furiou [((Bart_the_Mother.txt.gz, {tfidf=>7.525239E-4, idf
    =>1.7917595, tf=>4.199916E-4}), [603])]
49  futur [((Bart_the_Genius.txt.gz, {tfidf=>7.793647E-4, idf
    =>1.7917595, tf=>4.3497173E-4}), [300])]
50  pilot [((Bart_the_Fink.txt.gz, {tfidf=>8.3844614E-4, idf
    =>1.7917595, tf=>4.679457E-4}), [684])]
```

Appendix 2. Listing of BasicInvertedIndex

```
1  /**
2   * Basic Inverted Index
3   *
4   * This Map Reduce program should build an Inverted Index
        from a set of files.
5   * Each token (the key) in a given file should reference the
        file it was found
6   * in.
7   *
8   * The output of the program should look like this:
9   * sometoken [file001, file002, ... ]
10  *
11  * @author Kristian Epps
12  */
13 package uk.ac.man.cs.comp38120.exercise;
14
15 import java.io.IOException;
16 import java.util.Arrays;
17 import java.util.HashMap;
18 import java.util.StringTokenizer;
19
20 import org.apache.commons.cli.CommandLine;
21 import org.apache.commons.cli.CommandLineParser;
22 import org.apache.commons.cli.HelpFormatter;
23 import org.apache.commons.cli.OptionBuilder;
24 import org.apache.commons.cli.Options;
25 import org.apache.commons.cli.ParseException;
26 import org.apache.hadoop.conf.Configuration;
27 import org.apache.hadoop.conf.Configured;
28 import org.apache.hadoop.fs.Path;
29 import org.apache.hadoop.io.IntWritable;
30 import org.apache.hadoop.io.Text;
31 import org.apache.hadoop.mapreduce.Job;
32 import org.apache.hadoop.mapreduce.Mapper;
33 import org.apache.hadoop.mapreduce.Reducer;
34 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
35 import org.apache.hadoop.mapreduce.lib.input.FileSplit;
36 import org.apache.hadoop.mapreduce.lib.output.
       FileOutputFormat;
37 import org.apache.hadoop.util.Tool;
38 import org.apache.hadoop.util.ToolRunner;
39 import org.apache.log4j.Logger;
40
```

```
41 import uk.ac.man.cs.comp38120.io.array.ArrayListWritable;
42 import uk.ac.man.cs.comp38120.io.map.
       String2FloatOpenHashMapWritable;
43 import uk.ac.man.cs.comp38120.io.pair.PairOfWritables;
44 import uk.ac.man.cs.comp38120.io.triple.TripleOfIntsString;
45 import uk.ac.man.cs.comp38120.ir.Stemmer;
46 import uk.ac.man.cs.comp38120.ir.StopAnalyser;
47 import uk.ac.man.cs.comp38120.util.XParser;
48
49 public class BasicInvertedIndex extends Configured implements
       Tool
50 {
51     private static final Logger LOG = Logger
52             .getLogger(BasicInvertedIndex.class);
53
54     /*
55      * @returns A TripleOfIntsString in the format (TF,
            totalTokens, token)
56      */
57     public static class Map extends
58             Mapper<Object, Text, Text, PairOfWritables<
                TripleOfIntsString, ArrayListWritable<
                IntWritable>>>
59     {
60
61         // INPUTFILE holds the name of the current file
62         private final static Text INPUTFILE = new Text();
63
64         // TOKEN should be set to the current token rather
               than creating a
65         // new Text object for each one
66         @SuppressWarnings("unused")
67         private final static Text TOKEN = new Text();
68
69         // The StopAnalyser class helps remove stop words
70         @SuppressWarnings("unused")
71         private StopAnalyser stopAnalyser = new StopAnalyser
               ();
72
73         // The stem method wraps the functionality of the
               Stemmer
74         // class, which trims extra characters from English
               words
75         // Please refer to the Stemmer class for more
               comments
```

```
76          @SuppressWarnings("unused")
77          private String stem(String word)
78          {
79              Stemmer s = new Stemmer();
80
81              // A char[] word is added to the stemmer with its
                    length,
82              // then stemmed
83              s.add(word.toCharArray(), word.length());
84              s.stem();
85
86              // return the stemmed char[] word as a string
87              return s.toString();
88          }
89
90          // This method gets the name of the file the current
                Mapper is working
91          // on
92          @Override
93          public void setup(Context context)
94          {
95              String inputFilePath = ((FileSplit) context.
                    getInputSplit()).getPath().toString();
96              String[] pathComponents = inputFilePath.split("/"
                    );
97              INPUTFILE.set(pathComponents[pathComponents.
                    length - 1]);
98          }
99
100         /* Check if string has more than 1 uppercase letters,
                in that case
101          * leave the string in original form, else return it
                in lowercase.
102          * Example: David vs david (still understandable when
                lowercase)
103          * Example: DVD, NHL, BBC vs dvd, nhl, bbc
104          */
105         private String caseFolding(String token) {
106          Integer countOfUpperCaseLetters = 0;
107          for (Character c : token.toCharArray()) {
108           if (Character.isUpperCase(c))
                countOfUpperCaseLetters++;
109           if (countOfUpperCaseLetters > 1) return token;
110          }
111          return token.toLowerCase();
```

```
112         }
113
114         // This Mapper should read in a line, convert it to a
                set of tokens
115         // and output each token with the name of the file it
                was found in
116         public void map(Object key, Text value, Context
               context)
117                 throws IOException, InterruptedException
118         {
119          /* Get the filename from the first line */
120          String line = value.toString().trim();
121          HashMap<String, ArrayListWritable<IntWritable>> map
               = new HashMap<String, ArrayListWritable<
               IntWritable>>();
122          PairOfWritables<TripleOfIntsString,
               ArrayListWritable<IntWritable>> pair = new
               PairOfWritables<TripleOfIntsString,
               ArrayListWritable<IntWritable>>();
123          Text text = new Text();
124     IntWritable position = new IntWritable(0);
125     TripleOfIntsString keyTriple = new TripleOfIntsString();
126
127
128          /* Tokenize the text file */
129          StringTokenizer st = new StringTokenizer(line, "
                '\",;:.()[]{}!?/#$&^%  |=");
130          Integer tokenCount = st.countTokens();
131          if (tokenCount < 1) tokenCount = 1;
132
133          /* Iterate over set of tokens and output them with
                the key */
134          while (st.hasMoreTokens()) {
135           String token = st.nextToken();
136           position.set(position.get() + 1);;
137
138           /* Check if token is a stop word */
139           if (!StopAnalyser.isStopWord(token)) {
140
141            /* Do case folding */
142            token = caseFolding(token);
143
144            /* Stem the word */
145            token = stem(token);
146
```

```
147             /* Insert to map */
148             if (map.containsKey(token)) {
149              IntWritable p = new IntWritable(position.get());
150              map.get(token).add(p);
151             } else {
152              ArrayListWritable<IntWritable> pos = new
                    ArrayListWritable<IntWritable>();
153              IntWritable p = new IntWritable(position.get());
154              pos.add(p);
155              map.put(token, pos);
156             }
157            } /* end-if not stopword */
158           }
159
160           // http://lintool.github.io/MapReduceAlgorithms/
161
162           for (String s : map.keySet()) {
163            /* KeyTriple: (count of a particular token; count
                  of all tokents; file name) */
164            keyTriple.set(map.get(s).size(), tokenCount,
                  INPUTFILE.toString());
165            text.set(s);
166            pair.set(keyTriple, map.get(s));
167            context.write(text, pair);
168           }
169          }
170      }
171
172      public static class Reduce extends Reducer<Text,
173         PairOfWritables<TripleOfIntsString, ArrayListWritable<
               IntWritable>>,
174      Text,
175      ArrayListWritable<PairOfWritables<PairOfWritables<Text,
         String2FloatOpenHashMapWritable>, ArrayListWritable<
         IntWritable>>>>
176      {
177
178       Integer TOTAL_NUM_FILES = 6;
179
180       // This Reduce Job should take in a key and an iterable
             of file names
181          // It should convert this iterable to a writable
                array list and output
182          // it along with the key
183          public void reduce(
```

```
184                 Text key,
185                 Iterable<PairOfWritables<TripleOfIntsString,
                        ArrayListWritable<IntWritable>>> values,
186                 Context context) throws IOException,
                        InterruptedException
187         {
188
189          Integer countOfFilesTokenAppearsIn = 0;
190          /* The result array of all occurrences of the token
                */
191          ArrayListWritable<PairOfWritables<PairOfWritables<
                Text, String2FloatOpenHashMapWritable>,
                ArrayListWritable<IntWritable>>> occurences
192           = new ArrayListWritable<PairOfWritables<
                 PairOfWritables<Text,
                 String2FloatOpenHashMapWritable>,
                 ArrayListWritable<IntWritable>>>();
193
194          Double tf, idf;
195
196          /* Loop through the occurrences of a token */
197          for (PairOfWritables<TripleOfIntsString,
                ArrayListWritable<IntWritable>> pair : values) {
198           /* A copy of the Pair passed from the Mapper to
                 avoid Hadoop issues */
199              PairOfWritables<TripleOfIntsString,
                     ArrayListWritable<IntWritable>> pairCopy
200            = new PairOfWritables<TripleOfIntsString,
                  ArrayListWritable<IntWritable>>();
201           /* A copy of the Pair from the Pair passed from
                 Mapper to avoid Hadoop issues */
202              PairOfWritables<Text,
                     String2FloatOpenHashMapWritable>
                     nameAndMetaData
203            = new PairOfWritables<Text,
                  String2FloatOpenHashMapWritable>();
204           /* Result Pair that will be added to the
                 occurrences */
205              PairOfWritables<PairOfWritables<Text,
                     String2FloatOpenHashMapWritable>,
                     ArrayListWritable<IntWritable>> resultPair
206            = new PairOfWritables<PairOfWritables<Text,
                  String2FloatOpenHashMapWritable>,
                  ArrayListWritable<IntWritable>>();
207
```

```
208             countOfFilesTokenAppearsIn++;
209             pairCopy.set(pair.getLeftElement(), pair.
                    getRightElement());
210             String2FloatOpenHashMapWritable metadata = new
                    String2FloatOpenHashMapWritable();
211             Text filename = new Text();
212             filename.set(pairCopy.getLeftElement().
                    getRightElement());
213
214             tf   = (double) pairCopy.getLeftElement().
                    getLeftElement() / pairCopy.getLeftElement().
                    getMiddleElement();
215
216             /* Populate metadata */
217             metadata.put("tf", new Float(tf));
218             nameAndMetaData.set(filename, metadata);
219
220             /* Add result result */
221             resultPair.set(nameAndMetaData, pairCopy.
                    getRightElement());
222             occurences.add(resultPair);
223           }
224
225           idf  =  Math.log((double) TOTAL_NUM_FILES /
                    countOfFilesTokenAppearsIn);
226
227           /* Update the TF-IDF and IDF */
228           for (PairOfWritables<PairOfWritables<Text,
                    String2FloatOpenHashMapWritable>,
                    ArrayListWritable<IntWritable>> updatePair :
                    occurences) {
229             updatePair.getLeftElement().getRightElement().put("
                    tfidf", new Float(updatePair.getLeftElement().
                    getRightElement().get("tf") * idf));
230             updatePair.getLeftElement().getRightElement().put("
                    idf", new Float(idf));
231           }
232
233           /* Emit results */
234           context.write(key, occurences);
235         }
236       }
237
238       // Lets create an object! :)
239       public BasicInvertedIndex()
```

```
240     {
241     }
242
243     // Variables to hold cmd line args
244     private static final String INPUT = "input";
245     private static final String OUTPUT = "output";
246     private static final String NUM_REDUCERS = "numReducers";
247
248     @SuppressWarnings({ "static-access" })
249     public int run(String[] args) throws Exception
250     {
251
252         // Handle command line args
253         Options options = new Options();
254         options.addOption(OptionBuilder.withArgName("path").
                hasArg()
255                 .withDescription("input path").create(INPUT))
                    ;
256         options.addOption(OptionBuilder.withArgName("path").
                hasArg()
257                 .withDescription("output path").create(OUTPUT
                    ));
258         options.addOption(OptionBuilder.withArgName("num").
                hasArg()
259                 .withDescription("number of reducers").create
                    (NUM_REDUCERS));
260
261         CommandLine cmdline = null;
262         CommandLineParser parser = new XParser(true);
263
264         try
265         {
266             cmdline = parser.parse(options, args);
267         }
268         catch (ParseException exp)
269         {
270             System.err.println("Error parsing command line: "
271                     + exp.getMessage());
272             System.err.println(cmdline);
273             return -1;
274         }
275
276         // If we are missing the input or output flag, let
                the user know
277         if (!cmdline.hasOption(INPUT) || !cmdline.hasOption(
```

```
            OUTPUT))
278         {
279             System.out.println("args: " + Arrays.toString(
                    args));
280             HelpFormatter formatter = new HelpFormatter();
281             formatter.setWidth(120);
282             formatter.printHelp(this.getClass().getName(),
                    options);
283             ToolRunner.printGenericCommandUsage(System.out);
284             return -1;
285         }
286
287         // Create a new Map Reduce Job
288         Configuration conf = new Configuration();
289         Job job = new Job(conf);
290         String inputPath = cmdline.getOptionValue(INPUT);
291         String outputPath = cmdline.getOptionValue(OUTPUT);
292         int reduceTasks = cmdline.hasOption(NUM_REDUCERS) ?
                Integer
293                 .parseInt(cmdline.getOptionValue(NUM_REDUCERS
                        )) : 1;
294
295         // Set the name of the Job and the class it is in
296         job.setJobName("Basic Inverted Index");
297         job.setJarByClass(BasicInvertedIndex.class);
298         job.setNumReduceTasks(reduceTasks);
299
300         // Set the Mapper and Reducer class (no need for
                combiner here)
301         job.setMapperClass(Map.class);
302         job.setReducerClass(Reduce.class);
303
304         // Set the Output Classes
305         job.setMapOutputKeyClass(Text.class);
306         job.setMapOutputValueClass(PairOfWritables.class);
307         job.setOutputKeyClass(Text.class);
308         job.setOutputValueClass(ArrayListWritable.class);
309
310         // Set the input and output file paths
311         FileInputFormat.setInputPaths(job, new Path(inputPath
                ));
312         FileOutputFormat.setOutputPath(job, new Path(
                outputPath));
313
314         // Time the job whilst it is running
```

```
315         long startTime = System.currentTimeMillis();
316         job.waitForCompletion(true);
317         LOG.info("Job Finished in " + (System.
                currentTimeMillis() - startTime)
318                 / 1000.0 + " seconds");
319
320         // Returning 0 lets everyone know the job was
                successful
321         return 0;
322     }
323
324     public static void main(String[] args) throws Exception
325     {
326         ToolRunner.run(new BasicInvertedIndex(), args);
327     }
328 }
```