BDP: Homework 2: Executing Baum Welch Program

Due on Friday, December 22nd, 2017

Joonho Kwon

Hani Ramadhan - 201793254

Contents

1 Preliminary Informations

The purpose of this homework report is to show the success of executing BaumWelch algorithm [1] in hadoop using the source code located in https://github.com/DhruvKumar/Baum-Welch. This report is consisted of preparation, class diagram of the source code, and the execution result.

2 Preparation

This section will discuss about the source code provided for this project and setup for program's input file. The list of source code and its treatment is provided in Table 1 and Figure 1.

2.1 Source Code Modification

Source code modifications and reasons why some files are not used are briefly described below:

- **BaumWelchDriver.java** modification: Several options were not implemented in DefaultOptionCreator class. Thus, the options should be customized individually. This leads to make the model always built randomly at the beginning.
  - Removed all addOption(...) function calls that were not contained in DefaultOptionCreator:
    - addOption(DefaultOptionCreator.modelInOption()...
    - addOption(DefaultOptionCreator.numHiddenStates...
    - addOption(DefaultOptionCreator.numObservedStat...
  - Recreate removed all addOption() function call by adding own created DefaultOptionBuilder()...
  - Reassigned the appropriate getOption(...) commands as reflected in own created addOption() function calls.
  - Set buildRandom boolean value as true. Thus, the model always will be built randomly at initialization.
- **BaumWelchMapper.java** modification: There are some elements in BaumWelchConfigKeys.java which were not configured properly. They are *HIDDEN_STATES_MAP_PATH* and *EMITTED_STATES_MAP_PATH*. Hence, some modifications were applied.

Table 1: List of the BaumWelch source code files and their treatment

File Name	Treatment
BaumWelchCombiner.java	not used
BaumWelchConfigKeys.java	used, no modification
BaumWelchDriver.java	used, modified
BaumWelchDriver.java	not used
BaumWelchMapper.java	used, modified
BaumWelchModel.java	not used
BaumWelchReducer.java	used, no modification
BaumWelchUtils.java	used, no modification
IntArrayWritable.java	used, no modification
MapWritableCache.java	not used after modification of BaumWelchMapper.java

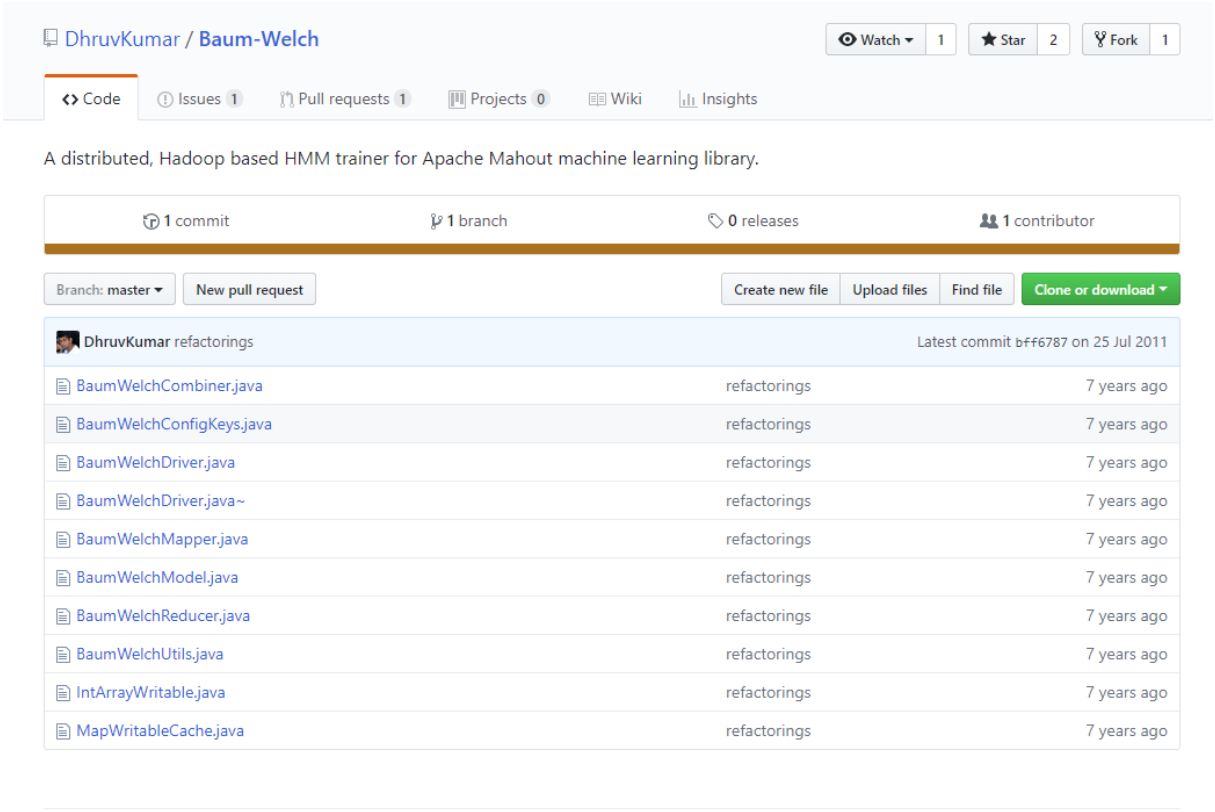

Figure 1: Source Code of Baum Welch Algorithm

  - Removal of *hiddenStatesWritableMap* and *emittedStatesWritableMap* variables. Which followed by removal to all of the related commands and variables. This also annuls the full usage/role of *MapWritableCache* in the project.

- **BaumWelchCombiner.java** unused: never called in the project
- **BaumWelchModel.java** unused: never called in the project
- **MapWritableCache.java** unused: already explained

2.2 BaumWelch Algorithm Input File

BaumWelch's input file requires sequence file(s) that should be put in input folder. Sequence file is a flat file which contains serialized key-value pair [2]. This flat file is commonly used in MapReduce task. BaumWelch program, as defined in BaumWelchMapper.java, needs LongWritable type as key and IntArrayWritable as value. Thus, the source code provided in https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/hadoop-sequence-file-example/ helped to initially create sequence file(s) as much as needed in BaumWelch input folder. The interaction of both programs are depicted in Figure 2.

3 Class Diagram

The class diagram of the BaumWelch program is presented in Figure 3, minus the BaumWelchModel class.

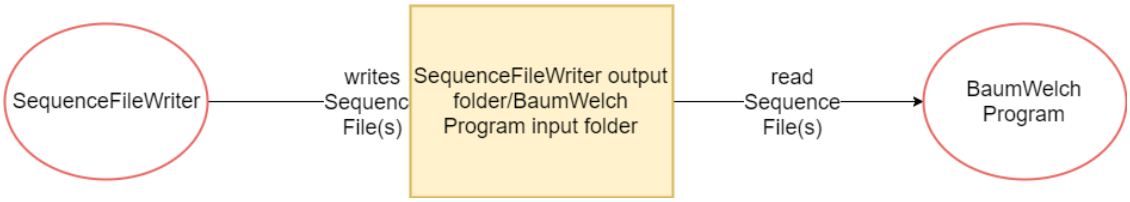

Figure 2: The interaction between Sequence File Writer and Baum Welch Program

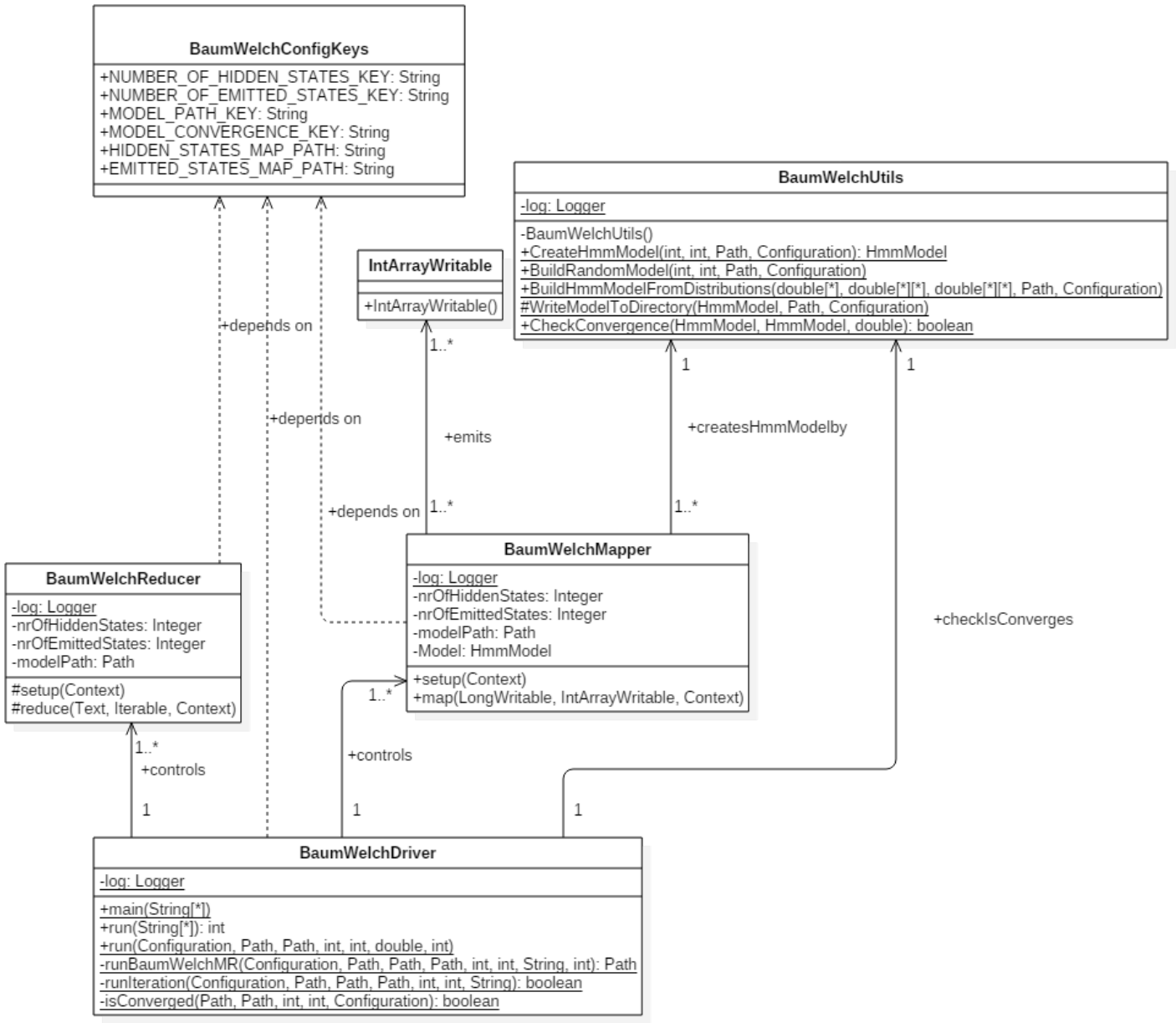

Figure 3: Class Diagram of Baum Welch program

```
hadoop jar BaumWelch.jar --input hani/input --output hani/output
--maxIter 10 --hmmModelInputPath hani/model --numHiddenStates 3
--numObservedStates 4
```

Figure 4: Example of BaumWelch program Execution, red colored means arguments

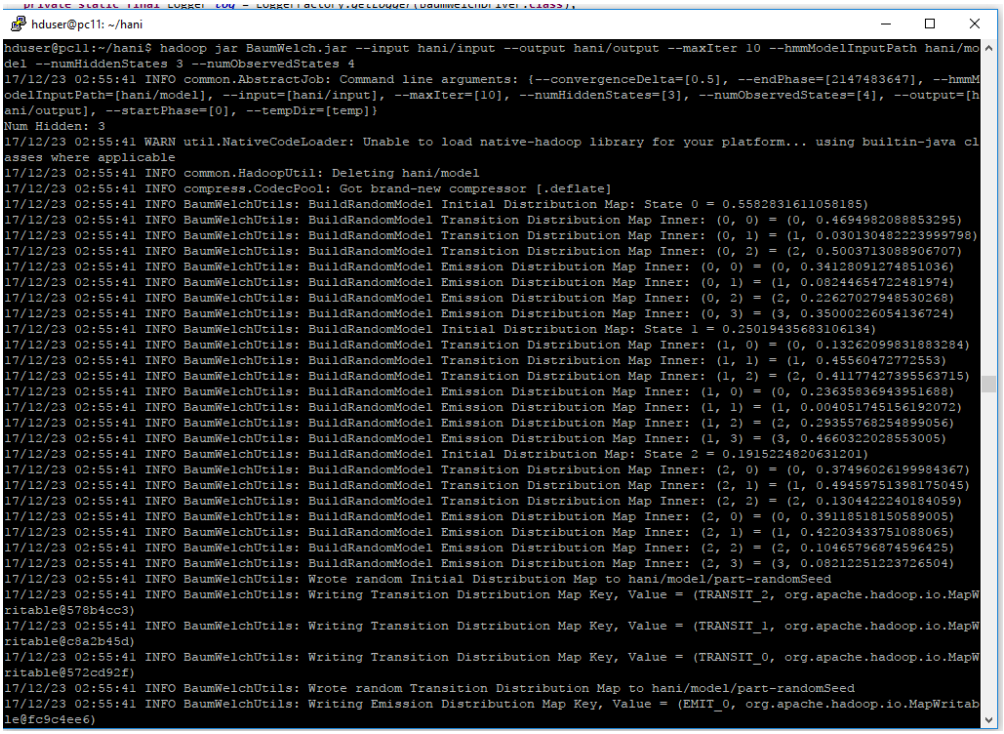

Figure 5: Start of of Baum Welch program execution

4 Program Execution

The example of the Baum Welch algorithm program execution is described in Figure 4. Options *–input*, *–output*, and *–maxIter* are built in from the program and cannot be neglected to run the program. The customized options are *–hmmModelInputPath*, *–numHiddenStates*, and *numObservedStates*. The options' information are detailed in Table 2. The screenshot of the start and the ending are presented in Figure 5 and Figure 6.

References

[1] D. Kumar, "Baum-Welch: A distributed, Hadoop based HMM trainer for Apache Mahout machine learning library.," Dec. 2015. original-date: 2011-07-24T16:35:10Z.

[2] R. Jhajj, "Hadoop Sequence File Example."

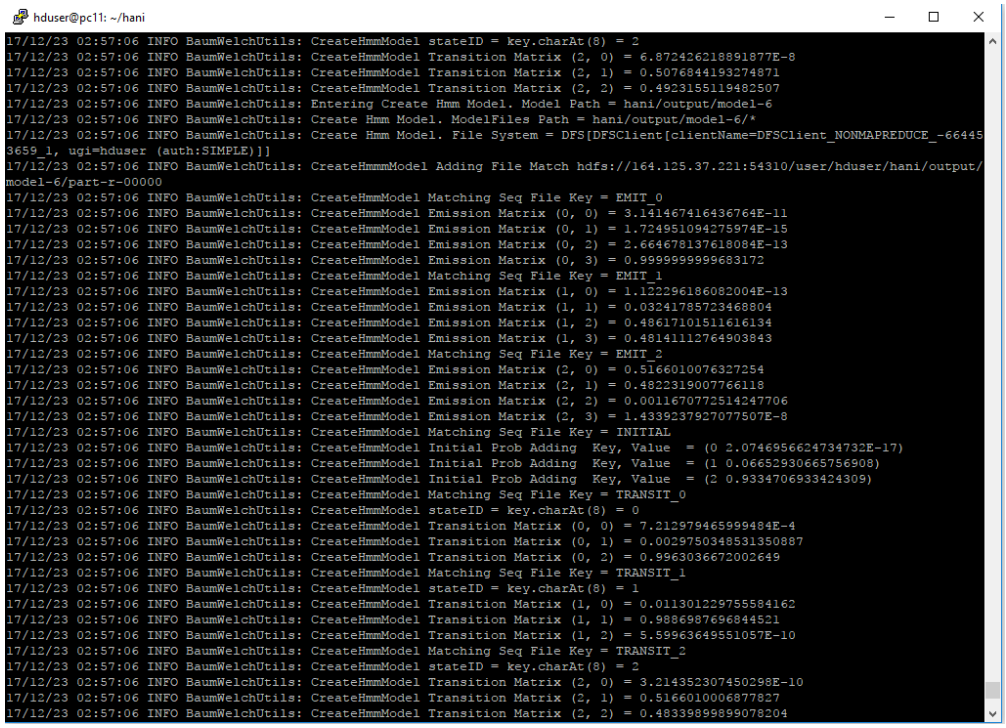

Figure 6: End of of Baum Welch program execution

Table 2: List of the BaumWelch program options

No	Option	Argument(s)	Description	Custom Created
1	–input (-i)	input	Path to job input directory.	No
2	–output (-o)	output	The directory path name for output.	No
3	–hmmModelInputPath (-hmmin)	hmmModelInputPath [hmmModelInputPath ...]	The input path of HMM Model.Must be of Sequence Filetype.	Yes
4	–numHiddenStates (-numHid)	numHiddenStates [numHiddenStates ...]	Number of Hidden States of HMM	Yes
5	–numObservedStates (-numObs)	numObservedStates [numObservedStates ...]	Number of Observed States of HMM	Yes
6	–convergenceDelta (-cd)	convergenceDelta	The convergence delta value. Default is 0.5	No
7	–maxIter (-x)	maxIter	The maximum number of iterations.	No
8	–help (-h)		Print out help	No
9	–tempDir	tempDir	Intermediate output directory	No
10	–startPhase	startPhase	First phase to run	No
11	–endPhase	endPhase	Last phase to run	No