Running LDA on Mahout under Hadoop

The following explains the usage of Apache Mahout's implementation of the Latent Dirichlet Allocation (LDA) learning algorithm. The idea behind the example is to extract given number of topics from a collection of text files located in a given directory. To run Mahout under Hadoop, make sure your Hadoop is running.

For Linux or Mac OS, set the following environment variables first:
JAVA_HOME, MAHOUT_HOME, HADOOP_HOME, HADOOP_CONF_DIR
using export in the .bash_profile file under the home directory.

The second step is to create the directory that will contain the text files we want to extract topics from and populate it.

1) Converting text documents to SequenceFile format

Mahout has a built in utility – sequdirectory to convert text files contained to a given directory to SequenceFile format:

The options of the command are as follows:

--input (-i) input	Path to job input directory.
--output (-o) output	The directory pathname for output.
--overwrite (-ow)	If present, overwrite the output directory before running job
--chunkSize (-chunk) chunkSize	The chunkSize in MegaBytes. Defaults to 64
--fileFilterClass (-filter) fileFilterClass	The name of the class to use for file parsing. Default: org.apache.mahout.text.PrefixAdditionFilter
--keyPrefix (-prefix) keyPrefix	The prefix to be prepended to the key
--charset (-c) charset	The name of the character encoding of the input file. Default to UTF-8
--help (-h)	Print out help
--tempDir tempDir	Intermediate output directory
--startPhase startPhase	First phase to run
--endPhase endPhase	Last phase to run

but the absolutely minimum to run the utility are (every command is executed from {mahout directory}/bin directory):

```
./mahout seqdirectory --input {full path to the folder containing text files}
--output {full path to folder that we want to save sequence files} -c UTF-8
```

the document id generated is {prefix}{relative path from parent}/document.txt

2) Creating vectors from SequenceFile

Again, Mahout has a built in utility – seq2sparse for this:

All options of the command are:

--minSupport (-s) minSupport	(Optional) Minimum Support. Default Value: 2
--analyzerName (-a) analyzerName	The class name of the analyzer
--chunkSize (-chunk) chunkSize	The chunkSize in MegaBytes. 100-10000 MB
--output (-o) output	The output directory
--input (-i) input	input dir containing the documents in sequence file format
--minDF (-md) minDF	The minimum document frequency. Default is 1
--maxDFPercent (-x) maxDFPercent	The max percentage of docs for the DF. Can be used to remove really high frequency terms. Expressed as an integer between 0 and 100. Default is 99.
--weight (-wt) weight	The kind of weight to use. Currently TF or TFIDF
--norm (-n) norm	The norm to use, expressed as either a float or "INF" if you want to use the Infinite norm. Must be greater or equal to 0. The default is not to normalize
--minLLR (-ml) minLLR	(Optional)The minimum Log Likelihood Ratio(Float) Default is 1.0
--numReducers (-nr) numReducers	(Optional) Number of reduce tasks. Default Value: 1
--maxNGramSize (-ng) ngramSize	(Optional) The maximum size of ngrams to create (2 = bigrams, 3 = trigrams, etc) Default Value:1
--overwrite (-ow)	If set, overwrite the output directory
--help (-h)	Print out help
--sequentialAccessVector (-seq)	(Optional) Whether output vectors should be SequentialAccessVectors. If set true else false
--namedVector (-nv)	(Optional) Whether output vectors should be NamedVectors. If set true else false
--logNormalize (-lnorm)	(Optional) Whether output vectors should be logNormalize. If set true else false

The minimum command to issue, continuing the example, is:

```
./mahout seq2sparse -i /home/kpzhang/Desktop/1 -o /home/kpzhang/Desktop/2 -wt t
```

3) Invoking LDA algorithm

With the documents prepared we can now invoke the LDA algorithm. Options to run the algorithm are:

--input (-i) input	Path to job input directory.
--output (-o) output	The directory pathname for output.
--overwrite (-ow)	If present, overwrite the output directory before running job
--numTopics (-k) numTopics	The total number of topics in the corpus
--numWords (-v) numWords	The total number of words in the corpus (can be approximate, needs to exceed the actual value)
--topicSmoothing (-a) topicSmoothing	Topic smoothing parameter. Default is 50/numTopics
--maxIter (-x) maxIter	The maximum number of iterations.
--help (-h)	Print out help
--tempDir tempDir	Intermediate output directory
--startPhase startPhase	First phase to run
--endPhase endPhase	Last phase to run

Continuing the example we issue the command:

```
./mahout lda -i /home/kpzhang/Desktop/2/tf-vectors -o /home/kpzhang/Desktop/3 -k 50 -v 200000
```

choosing to compute 50 topics in our corpus. The numWords parameter must exceed the total number of words in the previously computed dictionary. The easiest way to do this is to run it with a fictional number the first time, find this in the job initialization log outputted in the console

```
INFO: record buffer = 262144/327680
```

and rerun LDA with -v > 327680 in the example.

The input directory must point to the output directory of the previous face /tf-vectors or /tfidf-vectors depending on our previous choice.

4) Output the computed topics

After running LDA you can obtain an output of the computed topics using another Mahout utility - ldatopics:

All options of the command are:

```
--dict (-d) dict                Dictionary to read in, in the same format as one created by
                                org.apache.mahout.utils.vectors.lucene.Driver
--output (-o) output            Output directory to write top words
--words (-w) words              Number of words to print
--input (-i) input              Path to an LDA output (a state)
--dictionaryType (-dt) dictionaryType    The dictionary file type (text|sequencefile)
```

```
<span class="Apple-style-span" style="font-family: Georgia, 'Times New Roman',
'Bitstream Charter', Times, serif; font-size: 13px; line-height: 19px; white-space:
normal;">and to print out the topics of the example corpus we type:  </span>
```

```
mahout ldatopics -i /home/kpzhang/Desktop/3/state-9/ -d /home/kpzhang/Desktop/2/dictionary.*
 -o /home/kpzhang/Desktop/4 --dictionaryType sequencefile
```

Be careful here to use as the input directory the last state before convergence of the algorithm (sate-9 for the example). The output should be a set of files that each represent a computed topic and contain the words of that topic. Something like this:

```
end [p(end|topic_47) = 0.02054285450740693
class [p(class|topic_47) = 0.019173433558138983
you [p(you|topic_47) = 0.011081809083779152
ruby [p(ruby|topic_47) = 0.010939372894063723
code [p(code|topic_47) = 0.009912782417548813
pm [p(pm|topic_47) = 0.009565600712295945
07 [p(07|topic_47) = 0.008512880092545057
x [p(x|topic_47) = 0.007979504372069835
3 [p(3|topic_47) = 0.007552053207748147
from [p(from|topic_47) = 0.007501607551193776
your [p(your|topic_47) = 0.0074405804416695824
use [p(use|topic_47) = 0.007428773869438666
page [p(page|topic_47) = 0.007026473801340893
def [p(def|topic_47) = 0.006772484477458351
chapter [p(chapter|topic_47) = 0.006635243376794015
10 [p(10|topic_47) = 0.00600798063831032
have [p(have|topic_47) = 0.005999418317965101
2 [p(2|topic_47) = 0.005959277207972052
need [p(need|topic_47) = 0.005849514834292144
method [p(method|topic_47) = 0.005676919181434945
```