Sentence Generator

(Uses and implements tools from https://github.com/lintool/bespin.git)

**Github Repository:** git@github.com:ksayers91/SentenceGenerator.git

**Instructions and description:**

1. Open Terminal
2. Sentence generator is currently being hosted on Cloudlab's cloud computing services, and a working infrastuctre is available via ssh at:
   1. $ ssh -p 22 ksayers@ms0330.utah.cloudlab.us
   2. $ password:ksayers1
3. Once logged in navigate to the following directory.
   1. $ cd /local/hadoop/bespin
4. The sentence generator has a large text file corpus pre-loaded onto its file system that is shared between nodes.
5. Run the provided shell script: $ ./runSentenceGenerator.sh (wait several seconds)
6. The cluster will now begin to parse through a 1GB text file called the Brown Corpus, counting all unigram and bigrams. This part is done in parallel, which allows the system to achieve significantly greater performance.
7. After parsing the corpus, the program will move on to outputting all the counts to a text file. This text file is then used by a Java algorithm that evaluates probabilities of possible word occurrences and word pair occurrences that was found in the corpus.
8. Using this, sentences are generated using a weighted lottery selection system for word choices
9. After all probabilities are generated, the system will prompt you to enter a word to begin the sentence. Enter any word, that you would believe to appear in the corpus.
10. Once entered it will a generate sentence
11. Multiple sentences can be generated.
12. Entering '//s' will stop the program.
13. Output and input are stored the HDFS (shared storage space) which can be viewed at
    1. http://ms0330.utah.cloudlab.us:50070/
    2. Specifically output is located here: http://ms0330.utah.cloudlab.us:50070/explorer.html#/user/ksayers/DEMO/OUTPUT
14. https://github.com/lintool/bespin.git was used in the project as tools for parallel processing bigram counts on Apache Hadoop

**Bugs:**

- Input is not sanitized, as it was originally attended to accept files only. There will be certain input that causes errors.

**Issues & limitations:**

- Many features were still to be worked on. Including a better user interface and ability to pass whole files or file URLs.

- Only able to achieve about 50% parallelization. While still decent and more efficient, near full parallelization would have been nice and is achievable at some point with this specific implementation.

-Code is extremely dependent on the current environment set up on the Cloud lab cluster. It is not highly adaptable or reusable for others in its current state.

- Although Scala was planned to be a part of my implementation it did not make it into my final design due to time limitations. However, I did work with Scala when setting up and configuring Apache Spark. Files worked with can be seen at this location on the Master node:

/usr/local/spark/examples/src/main/scala/org/apache/spark/examples

-rw-r--r-- 1 ksayers shield-PG0 1553 Sep 28 18:03 HdfsTest.scala

-rw-r--r-- 1 ksayers shield-PG0 4647 Sep 28 18:03 LocalALS.scala

-rw-r--r-- 1 ksayers shield-PG0 2412 Sep 28 18:03 LocalFileLR.scala

-rw-r--r-- 1 ksayers shield-PG0 3446 Sep 28 18:03 LocalKMeans.scala

**User Friendliness:**

This project lacked a true User interface due to its nature in design. This project is more so aimed at being a tool used by developers. The user interface is solely command-line based interaction at the moment, but had plans to interact with a web interface which would allow a user to submit their own files, or links to files online. There is a user interface used by Hadoop, which has been referenced several times, for monitoring the system files and jobs. An additional user interface can be seen here: <http://ms0330.utah.cloudlab.us:19888/jobhistory/app> which provides a history of the jobs executed on the cluster. (You should see the one just previously executed).

**Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| Functionality Tested | Inputs | Expected Output | Actual Output |
| Submit File | Text file | System either responds with error message or proceeds to next page. | File submission is currently not fully implemented. Uses preexisting hosted file. |
| Download Bigram File | User Action | Bigram file is download, displaying all bigram counts and probabilities. | Prints to the Hadoop File system which is accessible through the Hadoop interface. |
| Download Unigram File | User Action | Unigram file is download, displaying all unigram counts and probabilities. | Prints to the Hadoop File system which is accessible through the Hadoop interface. |
| Generate Sentence | User Action | Random sentence is generated. Additionally, this can be used to determine probabilities are being used correctly. | Random sentences are generated as expected, with randomness and some coherent sentence structure. |
| Stress Testing | Big Data Text files (5-10GB) | Efficient high speed counting done within several minutes. | This was achieved relatively well. |

* Note: Most files that were used at one point for this project or residual as a part of testing, such as scripts and java files, are located throughout different folders within /local/Hadoop/ on the master node. (There are a lot). Unfortunately, parts of the project are 5-10gbs each due to testing data. All non-used files have been excluded from the github.
* SentenceGenerator project zip located online here: http://ms0330.utah.cloudlab.us:50070/explorer.html#/user/ksayers