# The Python and The Elephant
# Large Scale NLP with NLTK and Dumbo

**Nitin Madnani & Jimmy Lin**
{nmadnani, jimmylin}@umiacs.umd.edu

Presented at the 8th Annual Python Conference
on
February 20th, 2010

The simple non-hadoopified version of the word association task is fully described in the article "*Getting Started with Natural Language Processing on Python*" originally published in ACM Crossroads in 2007. An electronic version of the article is available at http://www.umiacs.umd.edu/~nmadnani/pdf/crossroads.pdf. Please follow the instructions in that article to run the serial version.

In order to run the parallelized, hadoopified version of the word association program, do the following:

1. Create an Amazon AWS account and set up the Amazon EC2 tools using the instructions described at this URL: http://docs.amazonwebservices.com/AmazonEC2/gsg/2007-01-03/. Please make sure that you have set up all you environment variables as described and put the ssh private key in the proper place.

2. Download the latest Cloudera Hadoop distribution from http://archive.cloudera.com/cdh/testing/ and unzip it somewhere, like under your home directory. Let `$HADOOPHOME` denote its path after unzipping.

3. Instead of using the default AMI (Amazon Machine Image) to be deployed on your instances, you will use an AMI that is built with both Hadoop and Dumbo. This is an AMI that I created and have made public. The AMI ID to use is `ami-d323ceba`. You will need to modify all the scripts under

`$HADOOPHOME/src/contrib/ec2` to use this AMI ID instead of the ID that it uses by default.

4. Once all the modifications have been made, you are ready to launch an EC2 cluster. To do so, run the following commands:

```
$> cd $HADOOPHOME/src/contrib/ec2/bin
$> ./hadoop-ec2 launch-cluster test-cluster 2
```

5. The above commands will launch a cluster called "test-cluster" with 1 master instance and 2 slave instances. The commands will take some time to finish and once they finish, they will print out the IDs of the launched instances. You can check that all the instances have started by using the command ec2-describe-instances at the shell. Once all the instances say "running" instead of "pending", you are ready to continue to the next step. (**Note**: If you wish to launch a cluster with 19 instances as I did for my talk, please replace the number 2 in the above command with 19. You cannot go any higher than that since the Amazon places a limit of 20 on the EC2 instances by default. To remove that limit, you will need to email Amazon.)
6. Log into the cluster as follows:

```
$> ./hadoop-ec2 login test-cluster
```

7. This will log you in as root into the cluster master instance. Once you are done, download the following two egg files onto the cluster by running these commands on the cluster master (**not** on your local machine).

```
$] wget http://nltk.googlecode.com/files/nltk-2.0b8-py2.6.egg

$] wget http://www.umiacs.umd.edu/~nmadnani/pycon/PyYAML.egg
```

8. Set up an easier to use alias for the 'hadoop' command:

```
$] alias hadoop='/usr/local/hadoop-0.20.1+152/bin/hadoop'
```

9. Copy over the ukWac corpus from S3 directly onto the Hadoop file system using the following command where MASTER_INTERNAL_IP is the internal

IP address for the master (the one that ends with '.internal' in the listing produced by ec2-describe-instances).

```
 $] hadoop distcp s3://ukwac-tagged hdfs://<MASTER_INTERNAL_IP>:
50001/
```

10. This will launch a job that can be tracked using the web-based interface to the hadoop jobtracker. This web-based interface can be found by opening this particular URL in your browser: `http://<MASTER_HOSTNAME>:50030`, where `<MASTER_HOSTNAME>` is the hostname of the cluster master instance. This hostname usually starts with '*ec2-*' and ends with '*.com*'.
11. Once the distcp job is complete, you are ready to run the actual word association python script on the cluster. Download it from my webpage :

```
 $] wget http://www.umiacs.umd.edu/~nmadnani/pycon/hwordassoc.py
```

12. Now, let's run the word association script using Dumbo:

```
 $] dumbo start hwordassoc.py -hadoop /usr/local/hadoop-0.20.1+152 \
-input /pycon-small -output /counts -libegg nltk-2.0b8-py2.6.egg \
-libegg PyYAML.egg -inputformat text -outputformat text \
-numReduceTasks 57 \
-hadoopconf stream.recordreader.compression=gzip
```

13. Let's go through the various options in that dumbo command:
    **'-hadoop'** : tells dumbo where the various hadoop jars are located.
    **'-input'** : the HDFS input directory.
    **'-output'** : the HDFS output directory where the outputs of the reducers will be stored.
    **'-libegg'** : tells dumbo which libraries to use when running the script. For this particular script, we are using the nltk and yaml libraries.
    **'-inputformat'** : tells dumbo that we are processing text files.
    **'-outputformat'** : tells dumbo to output plain text files, otherwise it will output in the typedbytes format.
    **'-numReduceTasks'** : tells dumbo to use 57 reduce tasks which is the same as the number of reducer slots available on a 19-slave cluster since there are 3 reducer slots available per slave. This is

usually the best option to maximize parallelism. [Note that we do **not** specify the number of mappers. Since the input files are in gzip format, hadoop streaming doesn't know how to split them and so it assigns each file to a separate mapper. This means that there will be 882 mappers in total. If the input files were not gzipped, this would ***not*** have been the case.]

**'-hadoopconf'** : tells dumbo to pass along a specific parameter into the actual hadoop streaming command line that dumbo will generate (it's just a wrapper, remember?). Here, the specific parameter to be passed is to tell hadoop streaming that the input files are in gzipped format.

14. Once you run this command, it will take about 5-6 hours for all the mappers and reducers to finish. You can keep checking the status using the web-based jobtracker as described above. Once that is done, you can see that the output directory 'counts' contains 57 files named '*part-00000*' through '*part-00056*'.

```
$] hadoop fs -ls /counts
```

15. You can transfer over those files to a local non-HDFS directory as follows:

```
$] mkdir /root/pycon-results
$] hadoop fs -get /counts/part-* /root/pycon-results
```

16. At this point, you can inspect the files and find the responses to any stimulus words you want. It might be cheaper to transfer the files to S3 since EC2 instances do not offer persistent storage and there is no point in paying for instance time just to analyze the output. You can find out how much this your cluster cost you by logging into your account and checking your account activity at http://aws.amazon.com.

And that's it! You have just finished running python on the cloud! If you have any questions or encounter any errors, please email me at nmadnani@umiacs.umd.edu.