Language Model Generation

170

In the previous part of the project, we generated the n-gram counts for a text corpus using a MapReduce job.

In this part we will compute a statistical language model using the n-gram counts and store them in an efficient manner so that they can be accessed easily from a web interface.

A statistical language model is a collection of probabilities of words appearing after a phrase. Using the n-gram counts as input, the probability of a word appearing after a phrase can be expressed in simple terms as:

$$\Pr(\text{word} \mid \text{phrase}) = \frac{\text{Count}(\text{phrase} + \text{word})}{\text{Count}(\text{phrase})}$$

As an example, consider the following input:

```
this                    1000
this is                 500
this is a               125
this is a blue          60
this is a blue house    20
```

The following probabilities can be calculated:

$$\Pr(\text{is} \mid \text{this}) = \frac{\text{Count}(\text{this is})}{\text{Count}(\text{this})} = \frac{500}{1000} = 0.5$$

$$\Pr(a \mid \text{this is}) = \frac{\text{Count}(\text{this is } a)}{\text{Count}(\text{this is})} = \frac{125}{500} = 0.25$$

Task

Your task is to generate the statistical language model for all words and phrases appearing in the n-gram counts generated from a text corpus. You must complete this task using a MapReduce job that reads the input file from HDFS and writes the output file to HBase. Specifically, you must:

1. Develop a schema in HBase to store the words appearing after phrases and their probabilities. You must also account for how the data is likely to be accessed from the user interface. Users will type a phrase and will expect to see an ordered list of the next word that the user is "most" likely to type.
2. Write a Mapreduce program to read the n-gram counts generated from the previous checkpoint, and process them to generate the probabilities as outlined above. The output from the MapReduce program should be written directly to an HBase table, following the schema that you have designed in step 1.

Note: For this checkpoint, assign the tag with Key: **`Project`** and Value: **`4.3`** for all resources

The information below will help you in these tasks, once you have completed building the language model, please proceed to the next page to connect it to the User Interface.

Hints, Assumptions and References:

1. You can use the ngrams generated from the previous checkpoint. If you need to re-create the ngrams, you may use **either** the Project Gutenberg dataset or the Wikipedia Text corpus, both of which are pre-processed and ready for use on S3. See the table below for locations.
2. If you are using the Wikipedia data set, we recommend that you directly load the dataset into HDFS in gzipped format and enable compression for inputs in your MapReduce program.
3. You can launch a cluster that has both HBase and Hadoop automatically using EMR.
4. You will want to ignore phrases that appear below a certain threshold, say **t**, from your n-gram count for your statistical language model to be accurate. **t** should be a command-line parameter to your MapReduce application. You may test your generator with **t=2** and then refine as necessary.
5. For a given phrase, store only the top **n** words with the highest probabilities. This value should also be a command-line parameter to your MapReduce application. You can start with **n=5**, and refine it as you see fit.
6. We recommend using `GenericOptionsParser` class, along with `apache.commons.cli` packages to parse command line options, but this is not necessary.
7. Use no more than 5 instances to complete the MapReduce job. Use spot pricing for instances that are larger than m1.small, if they are cheaper than the on-demand pricing.
8. HBase is covered in the Storage Module of the course; Practical aspects of HBase are available at http://hbase.apache.org/book/. *Hadoop, The Definitive Guide* by Tom White includes a good chapter on HBase. In addition, *HBase, The Definitive Guide* by Lars George is also a good reference.
9. As always, create a small test set to verify your approach and algorithm before running it over the entire dataset.
10. Once you have loaded the phrases, words and their associated probabilities into HBase, please use the `hbase shell` to test out some get operations.

Resources

Resource	Location
Gutenberg Dataset	`s3://15-319-s13/book-dataset/pg_00` to `pg_38`
Wikipedia Dataset	`https://s3.amazonaws.com/15-319-s13/wiki-dataset/part00.txt.gz` to `part63.txt.gz`