Your Paper

You

May 9, 2014

Abstract

Your abstract.

1 Apache Lucene

The basic raw code present on the Apache Lucene webpage indexes documents, but for the purposes of this project, we needed to index Tweets. Therefore, we modified the existing demo Indexer in order to come up with our own indexer for indexing tweets that are present across multiple files. In addition, we have written a two bash scripts that would make the process of indexing and search simpler. Assuming that we have already build Lucene, using ant, Here's how to invoke the scripts relating to indexing and searching.

```
./indexDocuments.sh <command> <InputDirectory>
```

This bash script is used to index tweets, present in multiple files. The description of arguments are as follows,

- **command** : can be either **compile** or **index**. In case the **command** equals **compile**, it compiles the java Indexer file. In this setting, the second parameter need not be given.

  In case the **command** equals **index**, the second parameter needs to be given. The second parameter is a directory that consists of files, which in turn have tweets within them.
- **InputDirectory** : Described above.

For the purposes of our project, we had nearly 1500 files, each containing 10K tweets. So following the steps above, we can index the tweets. Here's how we can perform search.

```
./searchDocuments.sh <command/queryFile>
```

This bash script is used to search a given query string against the tweets that we indexed in the previous step. The description of parameters for this script is as follows,

- **command/queryFile** : In case this parameter equals **compile**, this compiles the relevant java SearchDocument file. If we want to perform the search operation, then simply put in all the query strings in a file, and set the value of this parameter as the query file name.

2 Distributed LDA for Semantic Search

We wanted to build an Semantic Search Application, using Topic modelling keeping scalability as a primary design paradigm. We looked into multiple hadoop based LDA implementations, and finally decided on an implementation titled Mr.LDA.jar, here's the github link to the same.

`https://github.com/lintool/Mr.LDA`

2.1 Positives and Negatives of MrLDA: Our experiences

For performing Topic modelling on tweets, its necessary to do the same on Tweets, and not on the file containing the tweets. Unlike MrLDA, We found many LDA implementations, that performed topic modelling on documents, and not on each individual line of the document.

Performing LDA using MrLDA involved two steps. The first step was to preprocess the given input(remove stopwords, perform stemming, etc.). We faced a couple of problems; the most important being **setting the appropriate size of HADOOP HEAPSIZE variable**. We were using **4 m3.xlarge Amazon EMR machines**, and by trial and error we found that setting the heap size 20GB worked for us.

The second step was to do LDA training, but we found out that this step was extremely slow(**about 1% completion in 30 minutes with 4 m3.xlarge machines**). Despite our efforts, such as going on for bigger machines(with the same number of machines), we couldn't improve the speed greatly.

Despite the fact that it was a wonderful application, we had to discard it because of speed. And we decided to try out **Apache Mahout** suite of Machine-learning algorithms for LDA, especially the Collapsed Variational Bayes algorithm.

2.2 Apache Mahout : Our experiences

Performing Topic Modelling using Apache Mahout comprises of Five steps. We have listed down these, along with our modifications(if-any) and the innumerous versioning/compatibility issues that we faced and successfully/unsuccessfully overcame.