Finding Similar Items: Textually Similar Documents

Project Report (Group 25)

[avneesh@kth.se](mailto:avneesh@kth.se), [alggru@kth.se](mailto:alggru@kth.se) |ID2222 Data Mining | 12th November 2017

# Solution

In this project, we attempt to find similar documents among 10 documents using Jaccard Similarity. For this, we do the following:

1. Perform shingling of the documents by creating a set of unique shingles of length k (k-shingles) from the content of the document.
2. Then we map each shingle to an integer value using hashCode method of String class. Please note that as number of integer values can be less than possible k-shingles, hashing may result in collisions where two distinct shingles get mapped to the same integer value. But we assume that such collisions will be rare and will not significantly impact the overall Jaccard similarity.
3. Jaccard similarity between two documents (A and B) is computed by taking the ratio of unique shingles present in both documents and total distinct shingles in two documents.

For very large documents, space requirement to store k-shingles could be high and prohibitive. So instead, we employ Minhashing technique as follows:

1. Randomly generate n hash functions of the form ( (ax + b) mod c) by randomly generating n values for coefficients a and b. c is chosen to be a constant prime number.
2. For a given shingle set, minhash signature of size n is generated as follows. Each of n hash functions are applied to each element of given shingle set. And then minhash value which is the minimum hash value among all elements for a given hash function is selected. Thus, n minhash value set is generated for the given shingle set.
3. Finally, similarity between two documents is computed by calculating the ratio of identical minhash values among total n minhash values.

# Instructions to run

1. From the root folder run "sbt".
2. (Optional) In sbt shell run "eclipse" to create an eclipse project that can be imported into eclipse.
3. In sbt shell run "compile".
4. In sbt shell run "run".

Functionalities of various classes from the project are briefly described below

* Main.scala - Main entry point to the project
* Shingle.scala - Created hashed k-shingles of all documents present under src/main/resources/documents directory. k=10 defined in Main.scala
* CompareSet.scala - Computes Jaccard similarity of two given sets
* MinHashing.scala - Computes MinHash signature of a given size (n) for a given set of hashed shingles. n=8 used in Main.scala.
* CompareSignature.scala - Compares to given minHash signature set and returns the similarity measure (ratio of matching minhash values and total minhash values)

# Results