EPL451: Data Mining on the Web – Lab 9

University of Cyprus
Department of
Computer Science

Παύλος Αντωνίου

Γραφείο: B109, ΘΕΕ01

Apache Mahout Algorithms

Mahout currently provides the following algorithms:

- **Clustering (ομαδοποίηση)**
  - takes documents and groups topically related documents (unsupervised learning)
- **Classification (κατηγοριοποίηση)**
  - tries to assign documents to a correct category based on examples (supervised learning)
- **Recommendation (εισηγήσεις) - Lab 6**
  - takes user behavior and tries to predict items users might like

Apache Mahout for Clustering

- Plenty of Algorithms:
  - **k-Means [this lab]**, Fuzzy K-Means, Mean Shift, Canopy, Dirichlet process clustering, Spectral clustering.
- Notion of similarity in Mahout:
  - Euclidean Distance Measure
  - Squared Euclidean Distance Measure
  - Manhattan Distance Measure
  - Cosine Distance Measure
  - Tanimoto Distance Measure

Distance Measures

- Euclidean Distance Measure

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

- Squared Euclidean Distance Measure

$$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2$$

- Manhattan Distance Measure

$$d = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

- Cosine Distance Measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \ldots + a_n b_n)}{(\sqrt{(a_1^2 + a_2^2 + \ldots + a_n^2)}\sqrt{(b_1^2 + b_2^2 + \ldots + b_n^2)})}$$

- Tanimoto /Jaccard's Distance Measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \ldots + a_n b_n)}{\sqrt{(a_1^2 + a_2^2 + \ldots + a_n^2)} + \sqrt{(b_1^2 + b_2^2 + \ldots + b_n^2)} - (a_1 b_1 + a_2 b_2 + \ldots + a_n b_n)}$$

Task 1

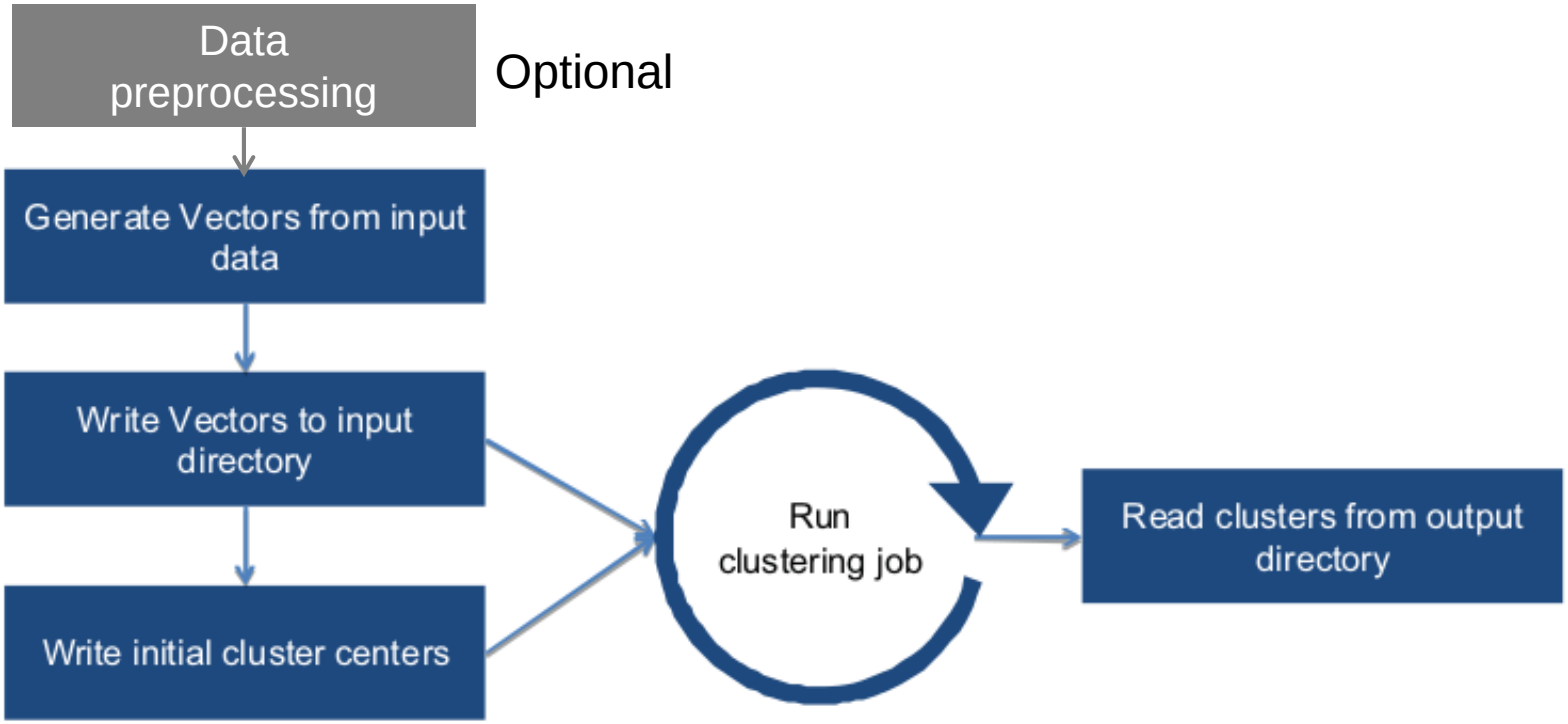
- Document clustering using Apache Mahout's k-Means
- Docs must be converted into vectors to be consumed by k-Means
- 3 ways to create Mahout vector representations:
  - Apache Lucene (or Apache Solr) index
  - Directory of text documents
    - Convert docs to SequenceFile format (Hadoop class)
  - Existing document vectors
  - For more info please see: http://mahout.apache.org/users/basics/creating-vectors-from-text.html

Task 1: Input dataset

- We will use data (articles) from Reuters, one of the largest international new agencies
- Download
  - http://www.cs.ucy.ac.cy/courses/EPL451/labs/LAB09/reuters21578.tar.gz
- Files are in Standard Generalised Markup Language (SGML) format
  - HTML and XML formats use SGML principles
- Each SGML file includes multiple articles from Reuters

Steps for Mahout Clustering

- There are some steps prior running Mahout clustering algorithms:



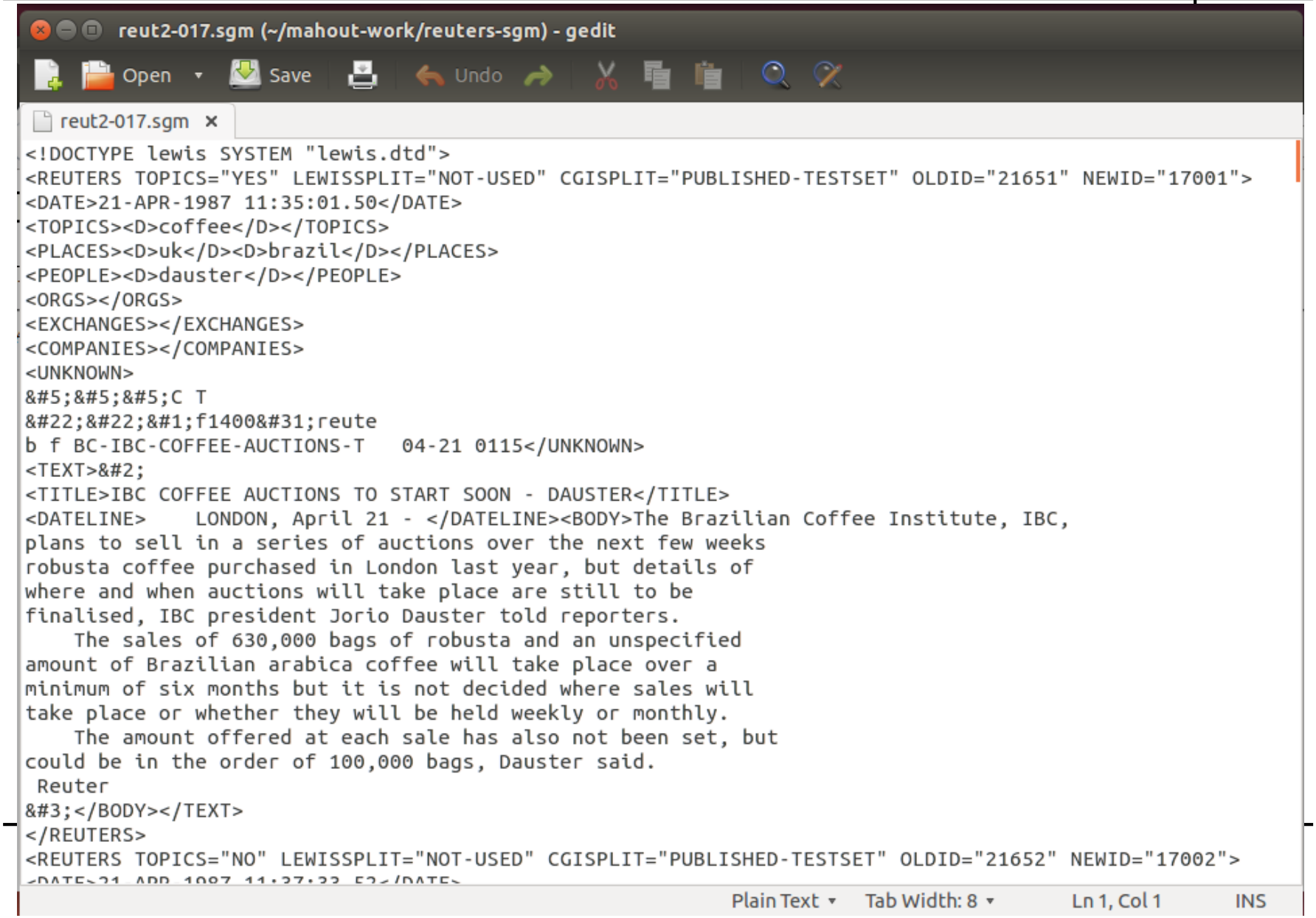
Steps in Task 1

- Data Preprocessing
  - Extract dataset, split each **SGML file** into multiple **simple text files**, each file contains one Reuters article; all files in the same directory
  - Convert each text file to a **sequence file** with <key, value> pairs
- Create **vector** (tf-idf) for each file and write in input directory (HDFS for this task)
- Choose and write cluster centers
- Run clustering algorithm (k-means for this task)
- Get the clustered data => find out which articles are clustered together (correlate, belong to the same cluster)

Extract Dataset

- Start Hadoop ☺
- Extract the SGML zip file
  - `cd`
  - `mkdir -p mahout-work/reuters-sgm`
  - `cd mahout-work/reuters-sgm`
  - `tar xzvf reuters21578.tar.gz`
  - Example SGML file
    `gedit mahout-work/reuters-sgm/reut2-017.sgm`

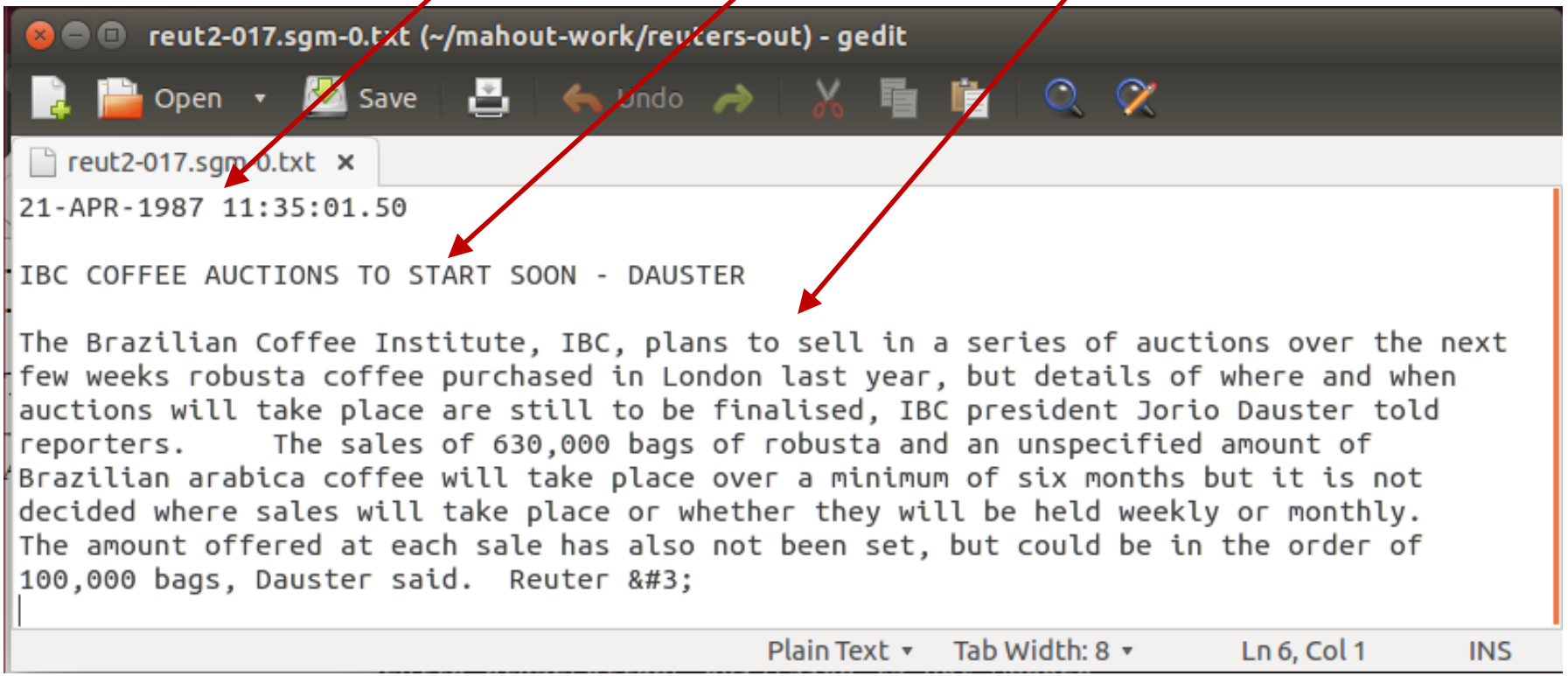
SGML file


reut2-017.sgm (~/mahout-work/reuters-sgm) - gedit
Open
Save
Undo
reut2-017.sgm
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS TOPICS="YES" LEWISSPLIT="NOT-USED" CGISPLIT="PUBLISHED-TESTSET" OLDID="21651" NEWID="17001">
<DATE>21-APR-1987 11:35:01.50</DATE>
<TOPICS><D>coffee</D></TOPICS>
<PLACES><D>uk</D><D>brazil</D></PLACES>
<PEOPLE><D>dauster</D></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;C T
&#22;&#22;&#1;f1400&#31;reute
b f BC-IBC-COFFEE-AUCTIONS-T 04-21 0115</UNKNOWN>
<TEXT>&#2;
<TITLE>IBC COFFEE AUCTIONS TO START SOON - DAUSTER</TITLE>
<DATELINE> LONDON, April 21 - </DATELINE><BODY>The Brazilian Coffee Institute, IBC,
plans to sell in a series of auctions over the next few weeks
robusta coffee purchased in London last year, but details of
where and when auctions will take place are still to be
finalised, IBC president Jorio Dauster told reporters.
The sales of 630,000 bags of robusta and an unspecified
amount of Brazilian arabica coffee will take place over a
minimum of six months but it is not decided where sales will
take place or whether they will be held weekly or monthly.
The amount offered at each sale has also not been set, but
could be in the order of 100,000 bags, Dauster said.
Reuter
&#3;</BODY></TEXT>
</REUTERS>
<REUTERS TOPICS="NO" LEWISSPLIT="NOT-USED" CGISPLIT="PUBLISHED-TESTSET" OLDID="21652" NEWID="17002">
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

Parse/Split SGML files

- Split each SGML file into multiple simple text files (each file => 1 article) containing **date, title, body**
- Parsing/Splitting can be done using Reuters parser given in the Lucene benchmark JAR file (bundled with Mahout)
  - `cd to **mahout** directory`
  - `bin/mahout org.apache.lucene.benchmark.utils.ExtractReuters ../mahout-work/reuters-sgm ../mahout-work/reuters-out`
  - `Example article file gedit mahout-work/reuters-out/reut2-017.sgm-0.txt`

Article file: DATE, TITLE, BODY



Hadoop Sequence Files

- Copy text files to HDFS
  - `hadoop fs -mkdir mahout-work`
  - `hadoop fs -mkdir mahout-work/reuters-out`
  - `hadoop fs -copyFromLocal ../mahout-work/reuters-out mahout-work/reuters-out`

- Convert directory of (small) text files to a directory of Hadoop SequenceFiles (binary files) using Map/Reduce processing

Hadoop Sequence Files

- Problems of small files:
  - Hadoop does not work very well with a lot of small files (smaller than HDFS block size)
    - memory overhead for the NameNode to hold huge amounts of small files
  - Also every map task processes a block of data at a time and when a map task has too little data to process, it becomes inefficient => starting up several such map tasks is an overhead
- Solution: Sequence Files
  - flat files containing key, value pairs
  - used as a container to store the small files
  - file metadata stored as key
  - file content as value
  - temporary outputs of maps are stored using SequenceFile

Hadoop Sequence Files

- Use seqdirectory tool to convert text files
  - `bin/mahout seqdirectory -i mahout-work/reuters-out -o mahout-work/reuters-out-seqdir -c UTF-8 -chunk 5`
- Input data (-i): text files in HDFS
- Output data (-o): binary key/value pairs in HDFS
  - Key: filename, Value: file content
- Chunk (-chunk): the chunkSize in MegaBytes.
- Check the generated sequence file
  - `bin/mahout seqdumper -i mahout-work/reuters-out-seqdir/part-m-00000 -n 4`
  - `bin/mahout seqdumper -i mahout-work/reuters-out-seqdir/part-m-00000 -c`

Vectors from Sequence Files

- Before running K-means algorithm, sequence files must be transformed to vectors
- Steps
  1. Compute Dictionary
     - list of words
  2. Assign integers for words
  3. Compute feature weights
     - E.g. number of word occurrences
  4. Create **vector for each document** using word-integer mapping and feature-weight
     - each word is one dimension

or

- Simply use:

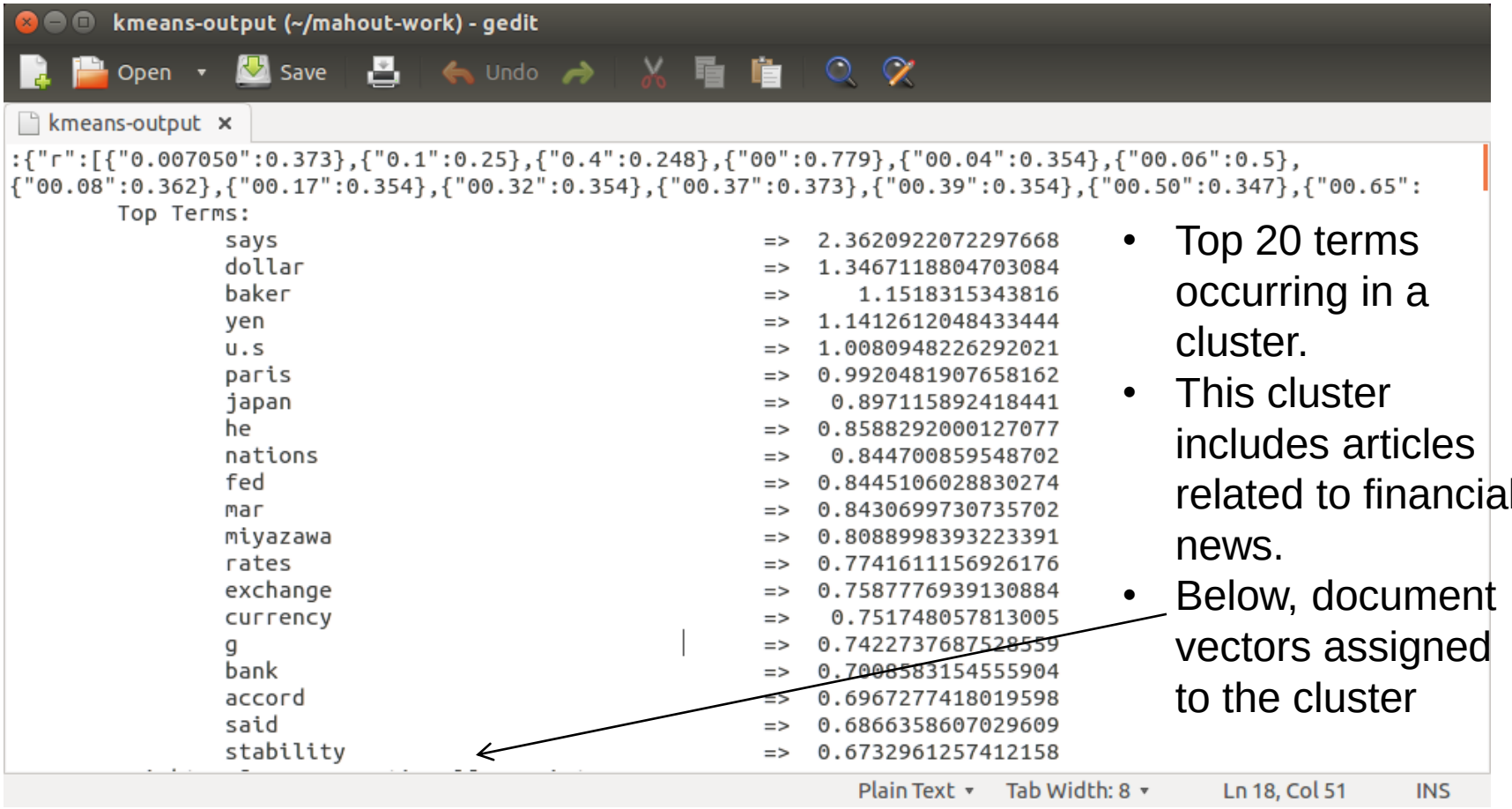
```
$ bin/mahout seq2sparse
```

Vectors from Sequence Files

- `bin/mahout seq2sparse -i mahout-work/reuters-out-seqdir/ -o mahout-work/reuters-out-seqdir-sparse-kmeans`
- There are several options for the seq2sparse command that may be required in other examples. A complete list can be found in the following link under: Creating Vectors from SequenceFile:
- http://mahout.apache.org/users/basics/creating-vectors-from-text.html

Vectors from Sequence Files

- ```
  bin/mahout seq2sparse \
      -i seqfiles/ \
      -o vectors/ \
      -a org.apache.lucene.analysis.core.WhitespaceAnalyzer \
      -ow -chunk 100 \
      -x 90 \    # default is 99
      -seq \
      -ml 50 \
      -n 2 \
      -nv
  ```

- -x 90 means that if a token appears in 90% of the documents it is considered a stop-word.
- -nv means that named vectors are returned
  ```

Vectors from Sequence Files

- The generated vectors dir should contain the following items:
  - reuters-vectors/df-count
  - reuters-vectors/dictionary.file-0
  - reuters-vectors/frequency.file-0
  - reuters-vectors/tf-vectors
  - **reuters-vectors/tfidf-vectors**
  - reuters-vectors/tokenized-documents
  - reuters-vectors/wordcount
- Examine by running:
  - `hadoop fs -ls mahout-work/reuters-out-seqdir-sparse-kmeans`

K-Means clustering

- `bin/mahout kmeans -i mahout-work/reuters-out-seqdir-sparse-kmeans/tfidf-vectors/ -c mahout-work/reuters-kmeans-clusters -cl -o mahout-work/reuters-kmeans -dm org.apache.mahout.common.distance.CosineDistanceMeasure -cd 0.1 -x 10 -k 20 -ow`
  - **-i < input vectors directory >**
  - -c < input clusters directory >
  - **-o < output working directory >**
  - -dm < Distance Measure technique >
  - -x < maximum number of iterations >
  - -k < optional number of initial clusters to sample from input vectors >
  - -cd < optional convergence delta. Default is 0.5 >
  - -ow < overwrite output directory if present >
  - -cl <run input vector clustering after computing Canopies>
  - Note: **if the -k argument is supplied, any clusters in the -c directory will be overwritten and -k random points will be sampled from the input vectors to become the initial cluster centers.**

K-Means clustering

INPUT

- `mahout-work/reuters-kmeans-clusters`
  - directory containing initial centroids of the clusters (if -k argument is supplied, this directory is overwritten)

OUPUT

- `mahout-work/reuters-kmeans/clusteredPoints`
  - a directory containing SequenceFile (IntWritable, WeightedVectorWritable) for clustered points (document vectors):
    - IntWritable *key* is the clusterId.
    - WeightedVectorWritable *value*
      - ***weight*** : probability that the vector is a member of the cluster
      - ***distance***: between cluster center and vector using chosen DistanceMeasure
      - **VectorWritable *vector***
  - `bin/mahout seqdumper -i mahout-work/reuters-kmeans/clusteredPoints/part-m-00000 -n 1`

K-Means clustering

OUPUTS

- `mahout-work/reuters-kmeans/clusters-*`
  - each directory contains SequenceFiles(Text, Cluster) produced by the algorithm for each iteration. The Text *key* is a cluster identifier string.
- `mahout-work/reuters-kmeans/clusters-*-final`
  - contains the final cluster details.

Inspect clusters

- Dump clusters to local filesystem
- bin/mahout clusterdump

```
-dt sequencefile                    # format {Integer => String}
-d mahout-work/reuters-out-seqdir-sparse-
kmeans/dictionary.file-*    # dictionary : {id => word}
-i mahout-work/reuters-kmeans/clusters-3-final
-o ../mahout-work/kmeans-output   # local filesystem
-p mahout-work/reuters-kmeans/clusteredPoints/
-b 100                              # format length
-n 20                               # number of top terms to print
```



- Top 20 terms occurring in a cluster.
- This cluster includes articles related to financial news.
- Below, document vectors assigned to the cluster

Inspect clusters

- Use of clusterdump utility. Options:
- -i input
  - The directory containing Sequence Files for the clusters
- -o output
  - The output file (in local file system). If not specified, dumps to the console
- -of outputFormat
  - The optional output format to write the results as. Options: TEXT, CSV, or GRAPH_ML
- -b substring
  - The number of chars of the asFormatString() to print
- -d dictionary
  - The dictionary file
- -dt dictionaryType
  - The dictionary file type (text|sequencefile)
- -dm distanceMeasure
  - The classname of the DistanceMeasure. Default is SquaredEuclidean.
- -n numWords
  - The number of top terms to print

Think about / Play with

- Number of iterations
- Convergence delta
- Distance measure

Issues

- How to get rid of useless (insignificant, common) words like a, the, …
  - Use StopwordsAnalyzer
  - In seq2sparse utility, option -x 90 means that if a token appears in 90% of the documents it is considered a stop-word
- How to choose appropriate weighting
  - If long text, go with tfidf. Use normalization if documents different in length

Issues

- How to scale
  - Use sampling apriori
  - Use small value of k to partially cluster data
  - And then do full clustering on each cluster.

Issues

- How to choose k
  - Figure out based on your data. Trial and error
  - Or use other preprocessing, maybe playing with distance thresholds
  - You can eliminate the guesswork by using the Canopy clusterer with appropriate distance thresholds to indicate the size of clusters.
    - ```
      bin/mahout canopy \
      -i reuters-vectors/tfidf-vectors \
      -o reuters-canopy-centroids \
      -dm org.apache.mahout.common.distance.EuclideanDistanceMeasure \
      -t1 1500 \ # points between 0 and t1 from centroid are included
      -t2 2000   # points between t1 and t2 from centroid are removed
      ```
    - After that run kmeans with canopy cluster centroids without -k parameter
    - You can also write your own clusterer
      ```

Issues

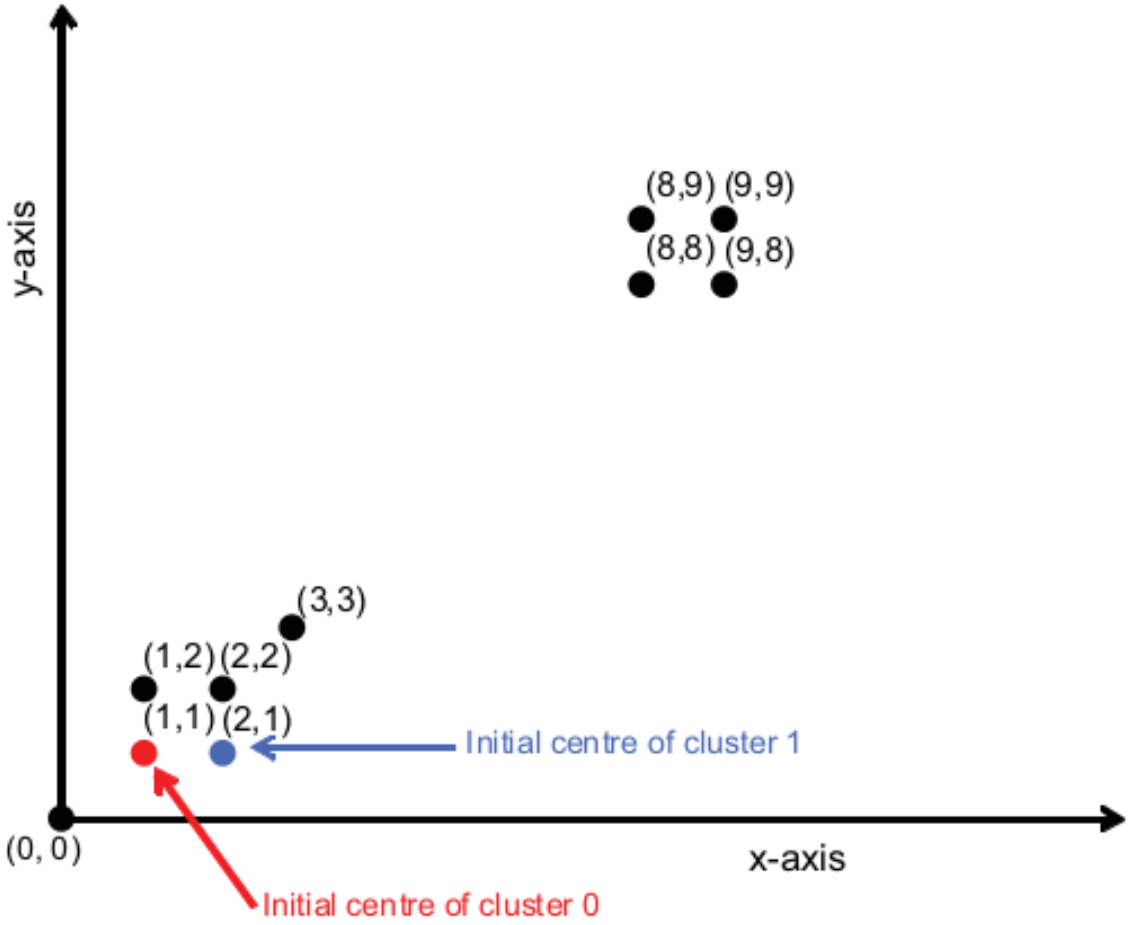
- How to improve Similarity Measurement
  - Not all features are equal
  - Small weight difference for certain types creates a large semantic difference
  - Use other measures : WeightedDistanceMeasure
  - Or write a custom DistanceMeasure

Tasks 2 & 3

- Write your own java program to cluster:
  - Task 2: a set of 2D points
  - Task 3: reuters's articles of Task 1
- Use Mahout k-Means

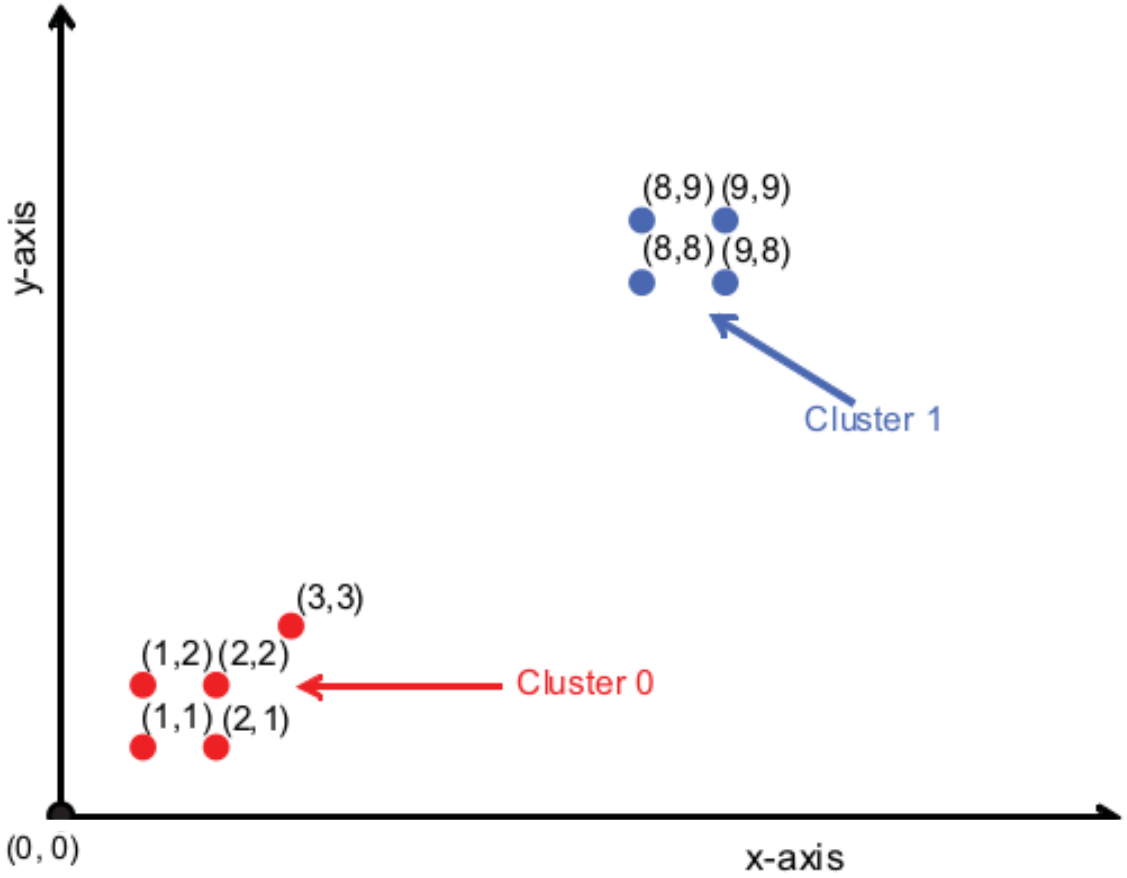
Task 2: Input data prior clustering

- Marking the initial clusters is an important step in k-means clustering



Task 2: Input data after clustering

- Even with distant centers, k-means algorithm is able to correctly iterate and re-evaluate centers based on the Euclidean Distance Measure



Playing around

Distance measure	Number of iterations	Vectors[a] in cluster 0	Vectors in cluster 1
EuclideanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
SquaredEuclideanDistanceMeasure	5	0, 1, 2, 3, 4	5, 6, 7, 8
ManhattanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
CosineDistanceMeasure	1	1	0, 2, 3, 4, 5, 6, 7, 8
TanimotoDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8

Task 2

- Download three (3) Java files implementing the simple k-means example presented before
  - related hadoop & mahout classes must be imported
  - use Apache Maven for managing dependencies

- Create Maven project
  - `mvn archetype:generate -DgroupId=com.csdeptucy.app -DartifactId=kmeans-clustering -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`

- Get into project folder
  - `cd kmeans-clustering`

Task 2 – Insights

- In order to run a k-means clustering job the method KMeansDriver.run() from Mahout API (version 0.7) is required to be invoked
-
```
public static void run(
org.apache.hadoop.conf.Configuration conf,
org.apache.hadoop.fs.Path input,
org.apache.hadoop.fs.Path clustersIn,
org.apache.hadoop.fs.Path output,
DistanceMeasure measure,
double convergenceDelta,
int maxIterations,
boolean runClustering,
double clusterClassificationThreshold,
boolean runSequential)
```

run() method parameters

- **input** - the directory pathname for input points
- **clustersIn** - the directory pathname for initial & computed clusters
- **output** - the directory pathname for output points
- **measure** - the DistanceMeasure to use
- **convergenceDelta** - the convergence delta value
- **maxIterations** - the maximum number of iterations
- **runClustering** - true if points are to be clustered after iterations are completed
- **clusterClassificationThreshold** - Is a clustering strictness / outlier removal parameter. Its value should be between 0 and 1. Vectors having pdf below this value will not be clustered.
- **runSequential** - if true execute sequential algorithm

Task 2: Unzip LAB09.zip

- Place 3 java files into /kmeans-clustering/src/main/java/com/csdeptucy/app folder
- Replace the pom.xml file
- Clean artifacts form the previous build and re-generate a jar file
  - `mvn clean package`
  - In case of `java.lang.OutOfMemoryError: Java heap space` error run in terminal:
    - `export MAVEN_OPTS=-Xmx1024m`
    - `mvn clean package`
- Run the application (start Hadoop first ☺)
  - `sudo java -cp target/kmeans-clustering-1.0-SNAPSHOT-jar-with-dependencies.jar com.csdeptucy.app.SimpleKMeansClustering`

Task 2 - Results

- 1.0: [1.000, 1.000] belongs to cluster 0
- 1.0: [2.000, 1.000] belongs to cluster 0
- 1.0: [1.000, 2.000] belongs to cluster 0
- 1.0: [2.000, 2.000] belongs to cluster 0
- 1.0: [3.000, 3.000] belongs to cluster 0
- 1.0: [8.000, 8.000] belongs to cluster 1
- 1.0: [9.000, 8.000] belongs to cluster 1
- 1.0: [8.000, 9.000] belongs to cluster 1
- 1.0: [9.000, 9.000] belongs to cluster 1

Task 3

- Use SimpleKMeansClustering.java file to create a new java program to cluster Reuters articles preprocessed in Task 1
- Reuters data in HDFS, “download” locally using:
  - `hadoop fs -copyToLocal /user/csdeptucy/mahout-work`
- The **input** points (as vectors) are stored in folder `mahout-work/reuters-out-seqdir-sparse-kmeans/tfidf-vectors`
- The **clustersIn** data (initial cluster centers) are stored in folder `mahout-work/reuters-kmeans-clusters/part-randomSeed`
- Make use of CosineDistanceMeasure() function
- For other params use the same values as in Task 1

Useful tips

- Sample:

  https://github.com/tdunning/MiA/tree/mahout-0.7

- Javadocs:
  - Hadoop:

    https://hadoop.apache.org/docs/r2.6.3/api/
  - Mahout:

    http://archive-primary.cloudera.com/cdh4/cdh/4/mahout-0.7-cdh4.3.2/mahout-core/

Apache Maven, Sequence Files

Appendix

Apache Maven

- Problem:
  - number of jar files (and their *dependencies*) must be downloaded and added to classpath
  - difficult to manually specify and discover all dependency libraries

- Solution: Apache Maven™
  - tool for building and managing any Java-based project
  - excellent dependency management mechanism
  - easy build process

Apache Maven

- Installation:
  - `sudo apt-get install maven`
- Create Maven project
  - `mvn archetype:generate -DgroupId=com.csdeptucy.app -DartifactId=applicationFolder -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`
- Get into project folder
  - `cd applicationFolder`
  - see project structure here
- POM.xml file
  - core of project's configuration

POM file example

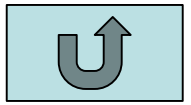
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Maven Quick Start Archetype</name>
    <url>http://maven.apache.org</url>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Maven phases

- Most common lifecycle **phases**:
  - **`validate`**: validate the project is correct and all necessary information is available
  - **`compile`**: compile the source code of the project
  - **`test`**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
  - **`package`**: take the compiled code and package it in its distributable format, such as a JAR
  - **`integration-test`**: process and deploy the package if necessary into an environment where integration tests can be run
  - **`verify`**: run any checks to verify the package is valid and meets quality criteria
  - **`install`**: install the package into the local repository, for use as a dependency in other projects locally
  - **`deploy`**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects
  - **`clean`**: cleans up artifacts created by prior builds
  - **`site`**: generates site documentation for this project
- Phases may be executed in sequence
  - `mvn clean package`

Test initial application

- Test the newly compiled and packaged JAR with the following command:
  - `java -cp target/applicationFolder-1.0-SNAPSHOT.jar com.csdeptucy.app.App`
- Which will print: `Hello World!`



Hadoop Sequence Files

- Sequence of Records, where each record is a <Key, Value> pair
  - SequenceFile provides *SequenceFile.Writer*, *SequenceFile.Reader* and *SequenceFile.Sorter* classes for writing, reading and sorting
- 3 SequenceFile Writers based on the *SequenceFile.CompressionType* used to compress key/value pairs:
  - Writer : Uncompressed records.
  - RecordCompressWriter : Record-compressed files, only compress values.
  - BlockCompressWriter : Block-compressed files, both keys & values are collected in 'blocks' separately and compressed. The size of the 'block' is configurable.

Writing your own Sequence Files

```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
Path path = new Path("testdata/part-00000");

SequenceFile.Writer writer = new SequenceFile.Writer(
        fs, conf, path, Text.class, Text.class);

for (int i = 0; i < MAX_DOCS; i++) {
        writer.append(new Text(documents(i).Id()),
        new Text(documents(i).Content()));
}
writer.close();
```