Serguey Khovansky
May 11, 2016

Frequency of Letters: How their usage has changed over thousand years

(application of Apache Flink technology)

Problem/Purpose:

This project examines the change in English letters usage over the last thousand years by computing frequencies with which they occur in several texts that have been written over that time period. In addition to frequencies, the project computes also beta in the regression of frequencies of one texts on the frequencies of another text. The examined texts were composed in both Old and Contemporary English. The work was implemented using Apache Flink technology.

Big Data Set:

Source of data: https://www.gutenberg.org

Title	Time of appearance	Author	Size
Beowulf	AD 1000		294k
The complete works	1600	William Shakespeare	5.3m
War and Peace	1870	Leo Tolstoy	3.2m
The Problems of Philosophy	1912	Bertrand Russell	263k

Format of data file: txt

Hardware:

Windows 7 running VMware with Linux CentOS 6

Software:

Technology/tools	Description
Apache Flink http://flink.apache.org (with DataSet API for Java)	Big data and streaming dataflow engine (competitor of Spark)
D3.js https://d3js.org/	JavaScript library for presenting and visualizing data
Java	Main tool to write code in Flink API
Apache Maven	Build manager for Java projects
Eclipse	Integrated development environment

Overview of steps:

1. Make sure Eclipse and Apache Maven are installed
2. Install Apache Flink
3. Consult Flink documentation (especially related to Data transformations) located in https://ci.apache.org/projects/flink/flink-docs-master/apis/batch/index.html
4. Write the code in Java. Note, that not all libraries are available in Java, e.g. Machine Learning is available only in Scala.
5. Write code to compute a Machine Learning exercise (i.e. some estimated parameters in a regression to understand how the frequencies change over time). Since Java ML library is still under construction, this task is quite time consuming.
6. Download data and run the code, get files with computed frequencies
7. Using D3.js plot the frequencies for different time epoch.
8. Wonder over the changes

Summary:

Flink is a very young technology that promises to outpace Spark. In contrast to Spark, which processes streaming data in batch mode, Flink processes streaming data in real time. However, being fledgling means not only high expectations, but also underdeveloped documentations and omission of important components (e.g. lack of proper Machine Learning Library written in Java along with its documentation). Its batch mode uses Data Transformation paradigm (similar to Spark) and its streaming mode (which nicely processes streaming data in real time) shows excellent examples with streaming from Twitter and Wiki. The part of Flink that works does so efficiently with low data latency. Although Flink technology is the most advantageous in processing streaming data, pedagogically its learning should be started with understanding batch mode as the documentation has a vast room for improvement.

YouTube URLs here:

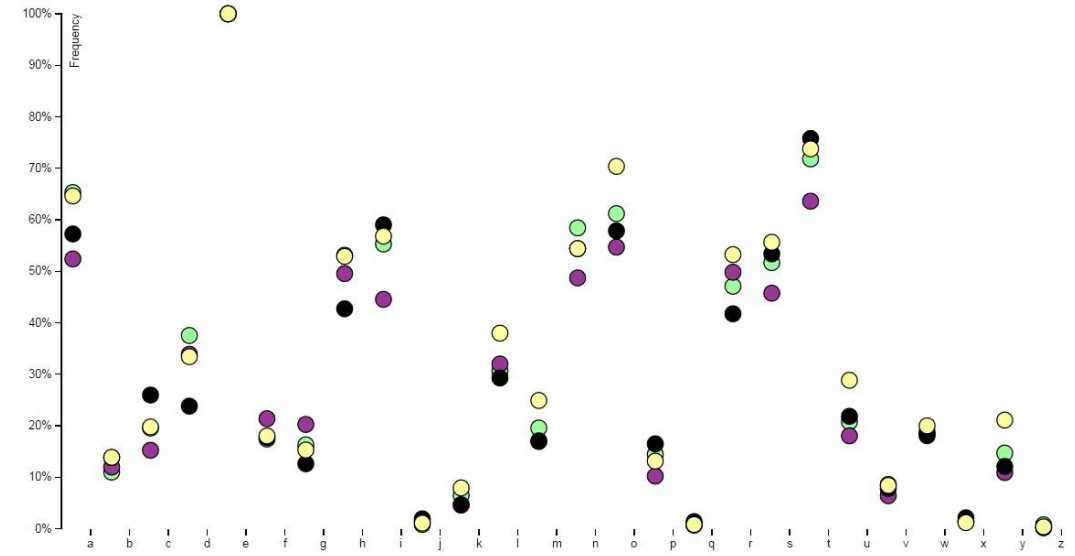
2-minute: https://youtu.be/DxaW3YLLTXM

15-minute:

Results

Frequencies of letters

Title	Time of appearance	Author	
Beowulf	AD 1000		
The Complete works	1600	William Shakespeare	
War and Peace	1870	Leo Tolstoy	
The Problems of Philosophy	1912	Bertrand Russell	



Regression results $Frequency_{text} = a + \beta \cdot Frequency_{\text{Beowulf}} + \varepsilon$, where estimate of $\beta = \frac{cov\ between\ Frequencies}{Variance\ of\ Beowulf\ Frequency}$

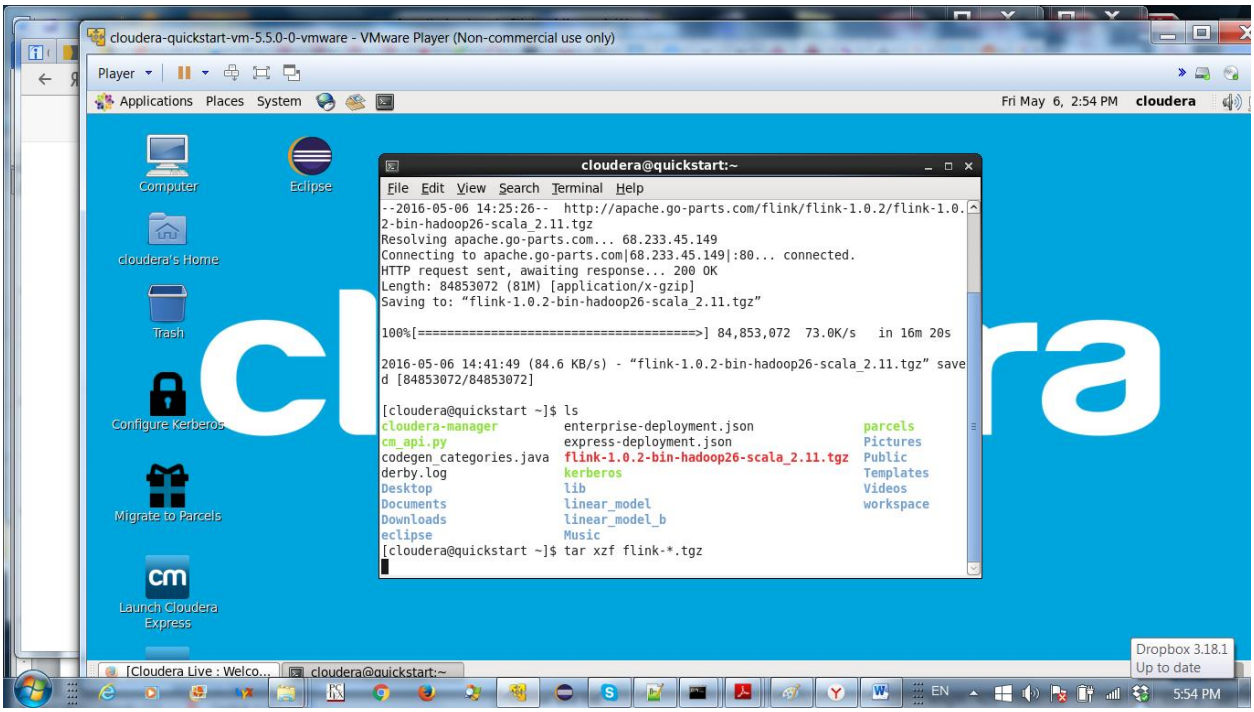
Author or Title – variable y	Time of appearance	Estimate of β	
Beowulf	AD 1000	1.000	
William Shakespeare	1600	1.075	
Leo Tolstoy	1870	1.066	
Bertrand Russell	1912	1.026	

Comment: In order to run *.html file with the picture, run the command **python -m http.server 8888 &** in the folder where the.html file is located.

Flink setup.

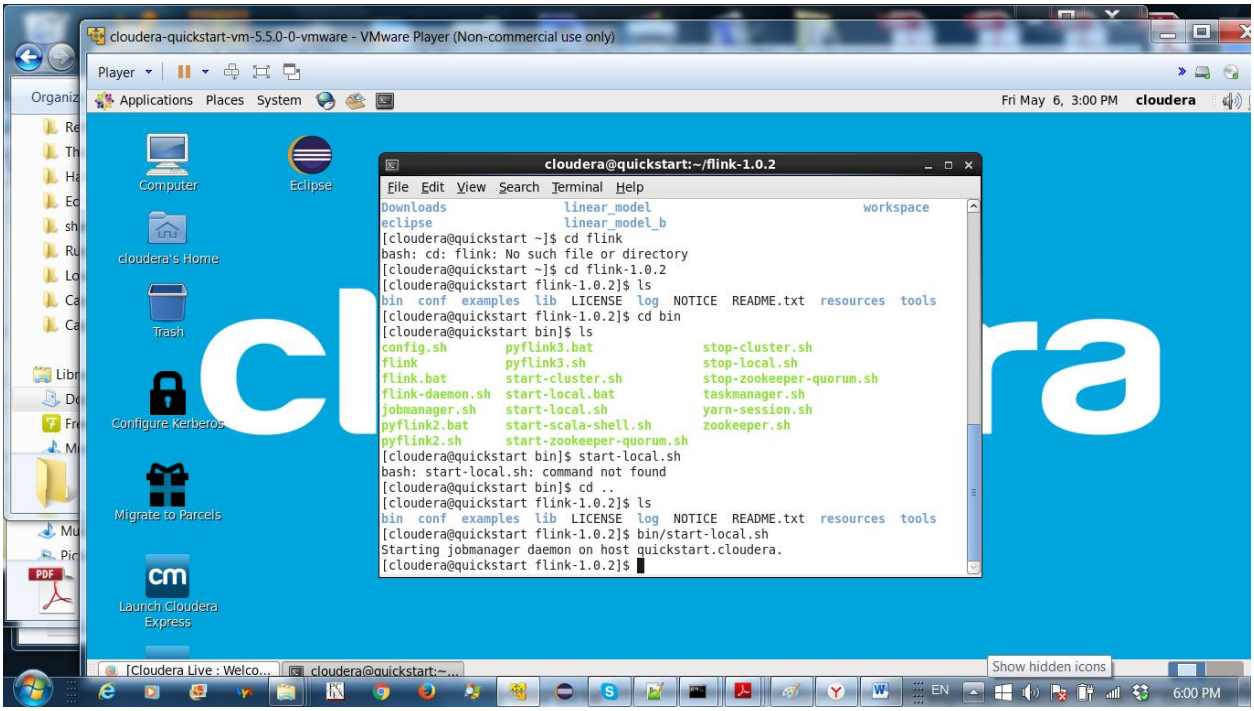
1. Check that Java is installed
2. Install Scala:
   wget http://downloads.typesafe.com/scala/2.11.1/scala-2.11.1.tgz
   tar xvf scala-2.11.1.tgz
   sudo mv scala-2.11.1 /usr/lib
   sudo ln -f -s /usr/lib/scala-2.11.1 /usr/lib/scala
   export PATH=$PATH:/usr/lib/scala/bin

3. Install Flink
   Download:
   wget http://apache.go-parts.com/flink/flink-1.0.2/flink-1.0.2-bin-hadoop26-scala_2.11.tgz



Unzip
tar xzf flink-*.tgz

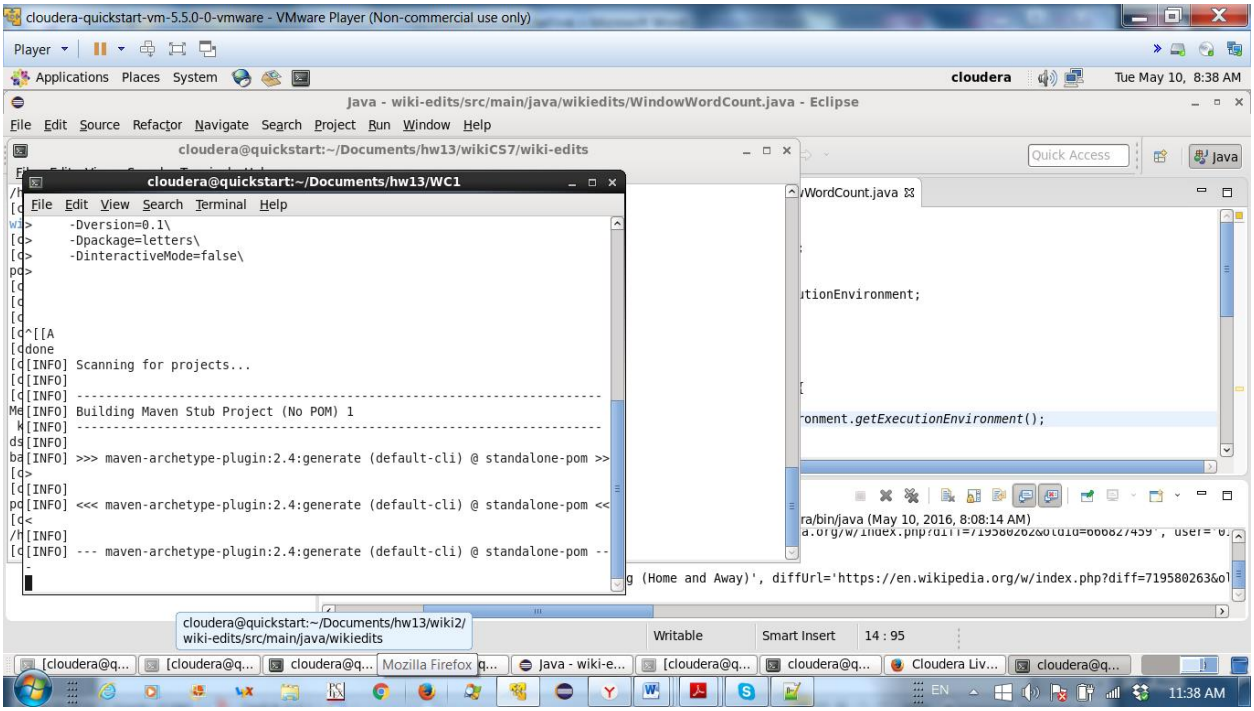
Run
[cloudera@quickstart flink-1.0.2]$ bin/start-local.sh



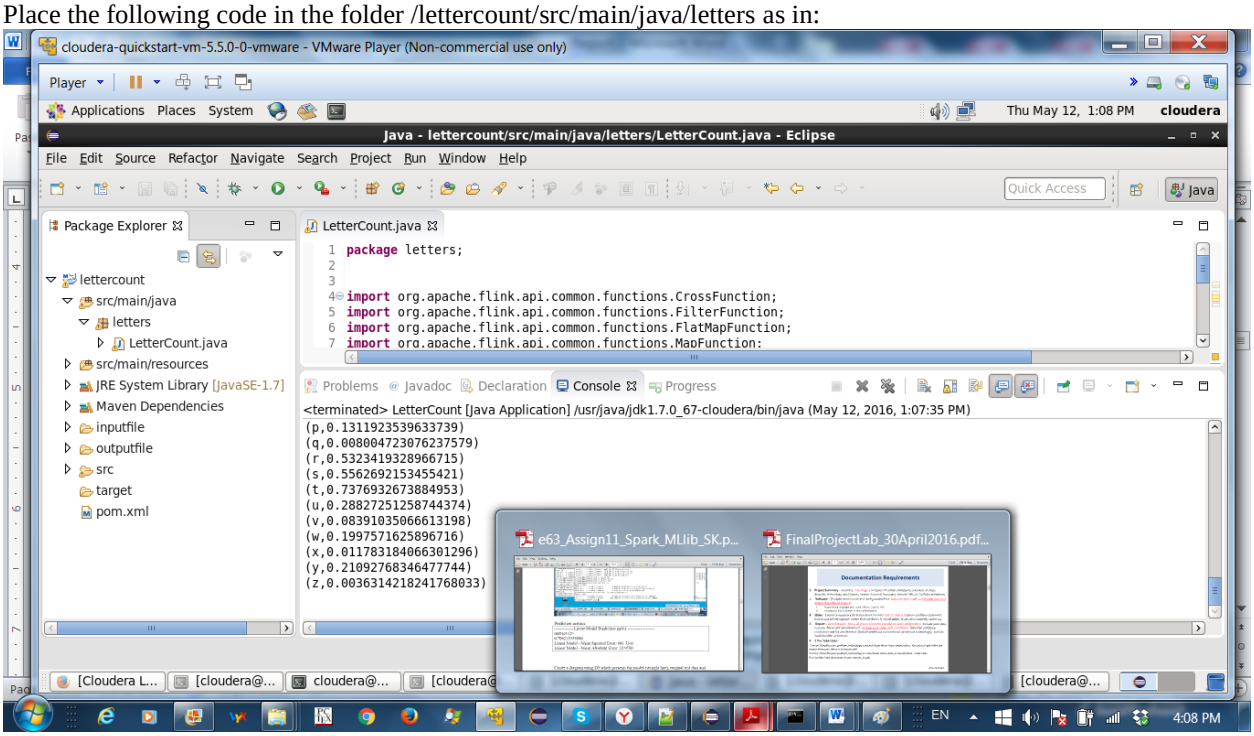
Details relevant to the current project.

1. Copy paste this Maven command in the root folder of the project

```
mvn archetype:generate\
  -DarchetypeGroupId=org.apache.flink\
  -DarchetypeArtifactId=flink-quickstart-java\
  -DarchetypeVersion=1.0.0\
  -DgroupId=lettercount\
  -DartifactId=lettercount\
  -Dversion=0.1\
  -Dpackage=letters\
  -DinteractiveMode=false\
```



Place the following code in the folder /lettercount/src/main/java/letters as in:



Several preliminary comments:

1. DataSet (or a DataStream if using stream) of Flink is equivalent to Spark's RDD
2. Flink job is executed lazily

```
/**
 * written by Serg Khovansky
```

```
* last change: May 2016S
* Using Flink technology, the code computes the number of letters in a given texts and compute regression beta in
* the regression Frequency of letters in text_y =a+beta* Frequency of letters in text_b
* Usage requires five input parameter: "
*                                                 + "1) path to input folder x  path_infile_x,"
*                                                 + "2) path to input folder y path_infile_y,"
*                                                 + "3) path to output folder  path_outfiles,"
*                                                 + "4) name of outfilename_x,"
*                                                 + "5) name of outfilename_y,"
*                                                 + "6) name of outfilename_beta_yx");
*
*
*/
package letters;


import org.apache.flink.api.common.functions.CrossFunction;
import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.common.functions.ReduceFunction;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.aggregation.Aggregations;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.util.Collector;



public class LetterCount {



        public static void main(String[] args) throws Exception {

                // set up the execution environment
                final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

                // get input  data folders for two texts (i.e. two different folders) and output folder
                if(args.length !=6){
                        System.err.println("Usage requires five input parameter: "
                                        + "1) path to input folder x  path_infile_x,"
                                        + "2) path to input folder y path_infile_y,"
                                        + "3) path to output folder  path_outfiles,"
                                        + "4) name of outfilename_x,"
                                        + "5) name of outfilename_y,"
                                        + "6) name of outfilename_beta_yx");
                        System.exit(-1);
                }
                String pathInputText_x = args[0];
                String pathInputText_y = args[1];
                String pathOutputText =  args[2];
                String outputfilename_x = args[3];
                String outputfilename_y = args[4];
                String outputfile_beta_xy = args[5];

                // output files name
                String outputFile_x = "file://" + pathOutputText+ outputfilename_x;
                String outputFile_y = "file://" + pathOutputText+outputfilename_y;
                String outputBeta = "file://" + pathOutputText+outputfile_beta_xy;


                //Work with two texts, hence, gets two DataSetsS
                DataSet<String> text_x = env.readTextFile(pathInputText_x);
                DataSet<String> text_y = env.readTextFile(pathInputText_y);


                //Get the splitter
    LineSplitter LS =  new LineSplitter();
                LS.set_mysplit("");
```

```
                    //The list of letters for which to compute frequencies
                    String abc = new String ("abcdefghijklmnopqrstuvwxyz");
                    final String[] abc_array = abc.split("");

                    // Start dataset transformations with text x
DataSet<Tuple2<String, Integer>> maxcounts2_x = text_x
      .flatMap(LS) // split the text into letters
                                        .filter(    // filter out everything else beyond letters
                                         new FilterFunction<Tuple2<String, Integer>>()
                                          {
                                                private static final long serialVersionUID = 1L;
                                                @Override
                                           public boolean filter(Tuple2<String, Integer> arg0) throws Exception
                                                {
                                                  for (String s:  abc_array){ if(s.equals(arg0.f0)) return true;}
                                                  return false;
                                           }
                                                }
                                         )
                  .groupBy(0)
                  .sum(1);

 // Start dataset transformations with text y
 DataSet<Tuple2<String, Integer>> maxcounts2_y = text_y
      .flatMap(LS) //  split the text into letters
                                        .filter(   // filter out everything else beyond letters
                                         new FilterFunction<Tuple2<String, Integer>>()
                                          {
                                                private static final long serialVersionUID = 1L;
                                                @Override
                                           public boolean filter(Tuple2<String, Integer> arg0) throws Exception
                                                {
                                                  for (String s:  abc_array){ if(s.equals(arg0.f0)) return true;}
                                                  return false;
                                           }
                                                }
                                         )
                  .groupBy(0) // group by the letter
                  .sum(1);  // sum up by the number of occurances

 //Find letter with max frequency
 DataSet<Tuple2<String, Integer>> maxcounts3_x = maxcounts2_x.aggregate(Aggregations.MAX, 1);
 DataSet<Tuple2<String, Integer>> maxcounts3_y = maxcounts2_y.aggregate(Aggregations.MAX, 1);



// compute relative frequency for both texts
                    DataSet<Tuple2<String, Double>>  relFreqCount_x = maxcounts2_x.cross(maxcounts3_x)
                                                .with(new ComputeRelFreq());//.sortPartition(1,Order.DESCENDING ) ;

                    DataSet<Tuple2<String, Double>>  relFreqCount_y = maxcounts2_y.cross(maxcounts3_y)
                                        // compute relative frequency
                                        .with(new ComputeRelFreq());//.sortPartition(1,Order.DESCENDING ) ;

                    /** Strat computing regression beta:
                     * Freq_y=a+beta*Freq_x+error, where e.g. Freq_y is frequency of occurance of letters in text x
                     * beta=cov(Freq_y,Freq_x)/var(Freq_x)
                    */
                    // compute sum of frequencies
                    DataSet<Tuple2<String, Double>>  relFreqCount_x_sum = relFreqCount_x.aggregate(Aggregations.SUM, 1);
                    DataSet<Tuple2<String, Double>>  relFreqCount_y_sum = relFreqCount_y.aggregate(Aggregations.SUM, 1);

                    // Compute total number of used letters, should be 26 in large English texts
                    DataSet<Tuple2<String, Double>> totalNumberLetters = relFreqCount_x.reduce(
                                        new ReduceFunction<Tuple2<String, Double>>()
                    {
                              private static final long serialVersionUID = 1L;

                              @Override
```

```
                                public Tuple2<String,  Double> reduce(Tuple2<String, Double> arg0, Tuple2<String, Double>
arg1)
                                                throws Exception {
                                        Tuple2<String, Double> out = new Tuple2<String, Double>();
                                        //arg1.f1=1D;
                                        out.f0 ="count";
                                        out.f1 = arg0.f1+1D;
                                        return out;
                                }
                                });

    //  compute average of relative Frequencies for text x
                        DataSet<Tuple2<String, Double>>
relFreqCount_x_average=relFreqCount_x_sum.cross(totalNumberLetters).with(
                                        new RatioFunc());
            //  compute average of relative Frequencies for text y
                        DataSet<Tuple2<String, Double>>
relFreqCount_y_average=relFreqCount_y_sum.cross(totalNumberLetters).with(
                                        new RatioFunc());
            //  compute deviation from averages of relative Frequencies for text x
                        DataSet<Tuple2<String, Double>>  relFreqCount_x_dev =
relFreqCount_x.cross(relFreqCount_x_average).with(
                                        new ComputeDeviation());
            //  compute deviation from averages of relative Frequencies for text y
                        DataSet<Tuple2<String, Double>>  relFreqCount_y_dev =
relFreqCount_y.cross(relFreqCount_y_average).with(
                                        new ComputeDeviation());


                        // make a DataSet with joint x and y frequencies (join on letters)
                        DataSet<Tuple2<Tuple2<String, Double>, Tuple2<String, Double>>>
                            relFreqCount_yx_dev = relFreqCount_y_dev.join(relFreqCount_x_dev)
                      .where(0)       // key of the first input (tuple field 0)
                      .equalTo(0);



     //Compute sum( x_dev^2)
                        DataSet<Tuple2<String, Double>> XX_dev = relFreqCount_x_dev
                                        .map(new MapFunction<Tuple2<String, Double>, Tuple2<String, Double>>(){
                                                        public Tuple2<String, Double>  map(Tuple2<String, Double> arg)
{
                                                                Tuple2<String, Double> out = new Tuple2<String,
Double>("",0D);
                                                                out.f1=arg.f1*arg.f1;
                                                                return out; }
                                        }
                                                        ).aggregate(Aggregations.SUM, 1);

                        //Compute sum( x_dev*y_dev)
                          DataSet<Tuple2<String, Double>>   XY_dev  =
                                          relFreqCount_yx_dev.
                                        map(new MapFunction< Tuple2< Tuple2<String, Double>, Tuple2<String, Double>>,
Tuple2<String, Double> >(){

                                                @Override
                                                public Tuple2<String, Double> map(Tuple2<Tuple2<String, Double>,
Tuple2<String, Double>> arg)
                                                                throws Exception {
                                                        Tuple2<String, Double> out = new Tuple2<String,
Double>("",0D);
                                                        out.f1 = arg.f0.f1*arg.f1.f1;
                                                        return out;
                                                }
                                }).aggregate(Aggregations.SUM, 1);


                          // compute beta of the regression
                        DataSet<Tuple2<String, Double>>  beta = XX_dev.cross(XY_dev)
                                        .with(new ComputeBeta());
```

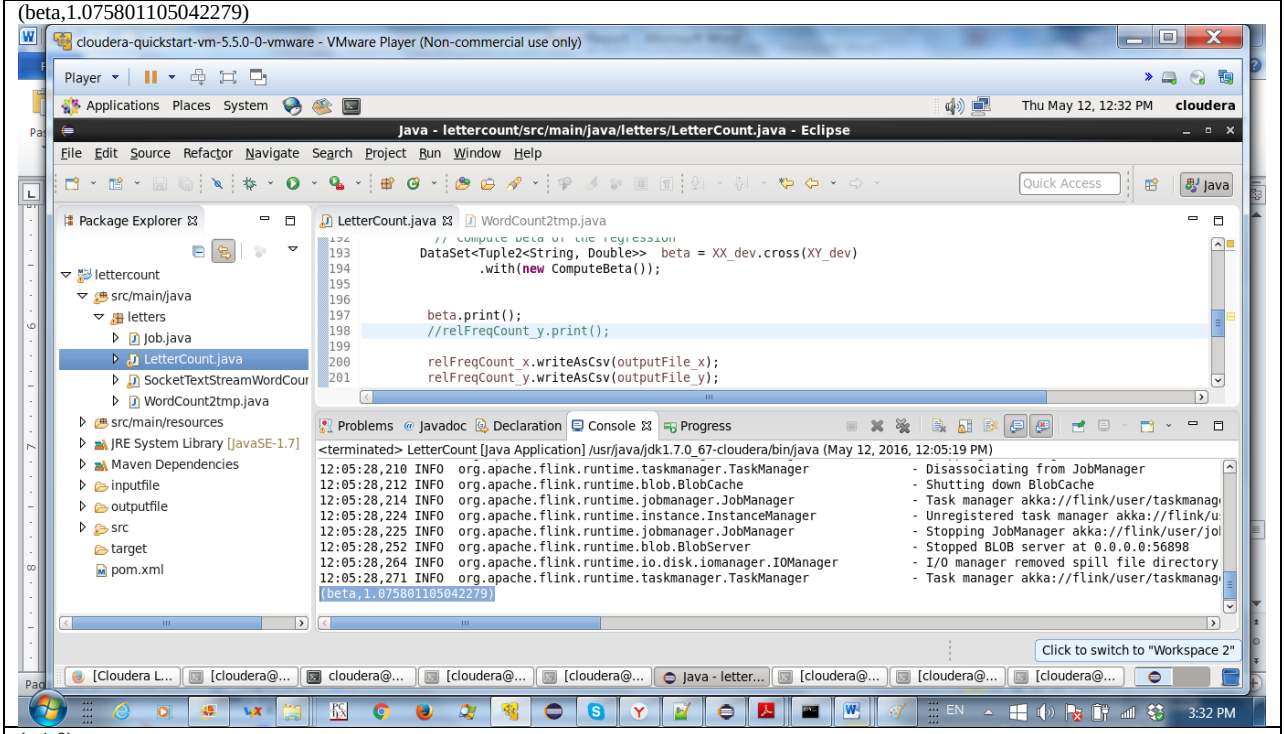
```
		beta.print();
		relFreqCount_y.print();

		relFreqCount_x.writeAsCsv(outputFile_x);
		relFreqCount_y.writeAsCsv(outputFile_y);
		beta.writeAsCsv(outputBeta);



	}

	//
	//		User Functions
	//


	 //Implements the Splitter
	public static class LineSplitter implements FlatMapFunction<String, Tuple2<String, Integer>> {
		DataSet<Tuple2<String, Integer>> A;
		Long myconst;
		String mysplit;
		LineSplitter(){mysplit="";}
		LineSplitter(String mysplit){this.mysplit= mysplit;}
		private static final long serialVersionUID = 1L;
		public void set_mysplit(String mysplit){
			this.mysplit=mysplit;
		}
		public void setTuple2(DataSet<Tuple2<String, Integer>> A){
			this.A=A;
			try {
				this.myconst= A.count();
			} catch (Exception e) {
				e.printStackTrace();
			}
		}

		@Override
		public void flatMap(String value, Collector<Tuple2<String, Integer>> out) {
			String[] tokens = value.toLowerCase().split(mysplit);

			// emit the pairs
			for (String token : tokens) {
				if (token.length() > 0) {
					Tuple2<String, Integer> TP = new Tuple2<String, Integer>(token, 1);
					out.collect(TP);
				}
			}
		}

	}

	// compute ratio of one data over the other data, need to normalize frequencies
	public static class ComputeRelFreq implements CrossFunction <Tuple2<String, Integer>, Tuple2<String, Integer>,
Tuple2<String,  Double>> {

		private static final long serialVersionUID = 1L;
		private double myscale;

		public ComputeRelFreq() {myscale=1L;}
		public void set_myscale(Long myscale){
			this.myscale = myscale;
		}

		public ComputeRelFreq(long myscale) {
			this.myscale = (double)myscale;
		}

		@Override
```

```
                        public Tuple2<String, Double> cross(Tuple2<String, Integer> A, Tuple2<String, Integer> B) throws Exception
        {
                                return new Tuple2<String, Double>(
                                                A.f0,
                                                (double)(myscale*(double)A.f1/ (double)B.f1) // relative frequency
                                );
                        }
                }

                // Compute Beta
                public static class ComputeBeta implements CrossFunction <Tuple2<String, Double>, Tuple2<String, Double>,
        Tuple2<String,  Double>> {

                        private static final long serialVersionUID = 1L;
                        public ComputeBeta() {}
                        @Override
                        public Tuple2<String, Double> cross(Tuple2<String, Double> arg0,
                                        Tuple2<String, Double> arg1) throws Exception {
                                return new Tuple2<String, Double>(
                                                "beta",
                                                arg1.f1/arg0.f1
                                                ) ;
                        }
                };

           // Compute ratio of two functions
                public static class RatioFunc implements CrossFunction <Tuple2<String, Double>, Tuple2<String, Double>,
        Tuple2<String,  Double>> {

                        private static final long serialVersionUID = 1L;
                        @Override
                        public Tuple2<String, Double> cross(Tuple2<String, Double> arg0,
                                         Tuple2<String, Double> arg1) throws Exception {

                                return new Tuple2<String, Double>(
                                                "Average",
                                                arg0.f1/arg1.f1
                                                ) ;
                        }
                };

                //ComputeDeviation()
                public static class ComputeDeviation implements CrossFunction <Tuple2<String, Double>, Tuple2<String, Double>,
        Tuple2<String,  Double>> {

                        private static final long serialVersionUID = 1L;

                        @Override
                        public Tuple2<String, Double> cross(Tuple2<String, Double> arg0,
                                         Tuple2<String, Double> arg1) throws Exception {
                                        return new Tuple2<String, Double>(
                                                arg0.f0,
                                                arg0.f1 - arg1.f1
                                                ) ;
                        }
                };


}// last one
```

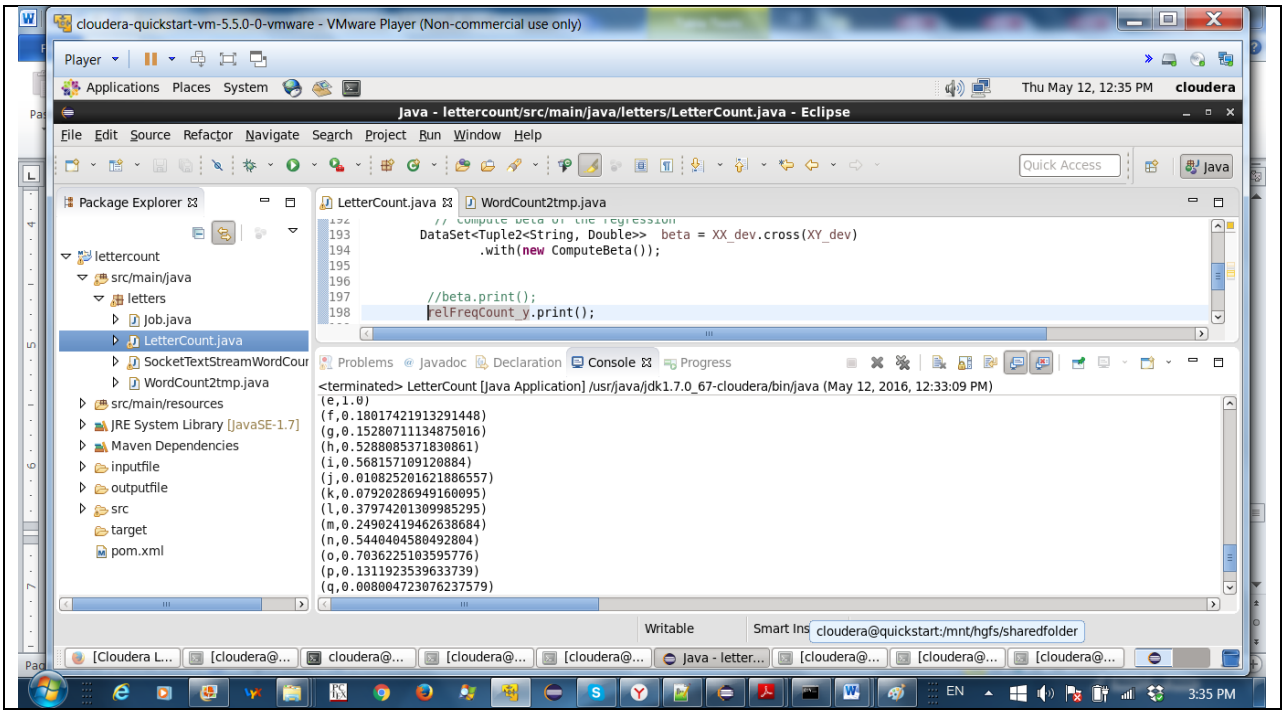
The pom.xml file can be found it the Appendix at the end of this document.

Sample output:

(beta,1.075801105042279)



(e,1.0)
(f,0.18017421913291448)
(g,0.15280711134875016)
(h,0.5288085371830861)
(i,0.568157109120884)
(j,0.010825201621886557)
(k,0.07920286949160095)
(l,0.37974201309985295)
(m,0.24902419462638684)
(n,0.5440404580492804)
(o,0.7036225103595776)
(p,0.1311923539633739)
(q,0.008004723076237579)
(r,0.5323419328966715)
(s,0.5562692153455421)
(t,0.7376932673884953)
(u,0.28827251258744374)
(v,0.08391035066613198)
(w,0.1997571625896716)
(x,0.011783184066301296)
(y,0.21092768346477744)
(z,0.0036314218241768033)


cloudera-quickstart-vm-5.5.0-0-vmware - VMware Player (Non-commercial use only)
Player
Applications Places System
Thu May 12, 12:35 PM cloudera
Java - lettercount/src/main/java/letters/LetterCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
LetterCount.java
WordCount2tmp.java
lettercount
src/main/java
letters
Job.java
LetterCount.java
SocketTextStreamWordCour
WordCount2tmp.java
src/main/resources
JRE System Library [JavaSE-1.7]
Maven Dependencies
inputfile
outputfile
src
target
pom.xml
193 DataSet<Tuple2<String, Double>> beta = XX_dev.cross(XY_dev)
194 .with(new ComputeBeta());
197 //beta.print();
198 relFreqCount_y.print();
Problems Javadoc Declaration Console Progress
<terminated> LetterCount [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (May 12, 2016, 12:33:09 PM)
(e,1.0)
(f,0.18017421913291448)
(g,0.15280711134875016)
(h,0.5288085371830861)
(i,0.568157109120884)
(j,0.010825201621886557)
(k,0.07920286949160095)
(l,0.37974201309985295)
(m,0.24902419462638684)
(n,0.5440404580492804)
(o,0.7036225103595776)
(p,0.1311923539633739)
(q,0.008004723076237579)
Writable
Smart Ins
cloudera@quickstart:/mnt/hgfs/sharedfolder
3:35 PM

Appendix:

pom.xml

```
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->
<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>

	<version>1.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>Flink Quickstart Job</name>
	<url>http://www.myorganization.org</url>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<flink.version>1.0.0</flink.version>
	</properties>

	<repositories>
		<repository>
			<id>apache.snapshots</id>
			<name>Apache Development Snapshot Repository</name>
			<url>https://repository.apache.org/content/repositories/snapshots/</url>
			<releases>
				<enabled>false</enabled>
			</releases>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
	</repositories>

	<!--

		Execute "mvn clean package -Pbuild-jar"
		to build a jar file out of this project!

		How to use the Flink Quickstart pom:

		a) Adding new dependencies:
			You can add dependencies to the list below.
			Please check if the maven-shade-plugin below is filtering out your dependency
			and remove the exclude from there.

		b) Build a jar for running on the cluster:
			There are two options for creating a jar from this project
```

```
                        b.1) "mvn clean package" -> this will create a fat jar which contains all
                                        dependencies necessary for running the jar created by this pom in a cluster.
                                        The "maven-shade-plugin" excludes everything that is provided on a running Flink
cluster.

                        b.2) "mvn clean package -Pbuild-jar" -> This will also create a fat-jar, but with much
                                        nicer dependency exclusion handling. This approach is preferred and leads to
                                        much cleaner jar files.
        -->

        <dependencies>
                <dependency>
                        <groupId>org.apache.flink</groupId>
                        <artifactId>flink-java</artifactId>
                        <version>${flink.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.apache.flink</groupId>
                        <artifactId>flink-streaming-java_2.10</artifactId>
                        <version>${flink.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.apache.flink</groupId>
                        <artifactId>flink-clients_2.10</artifactId>
                        <version>${flink.version}</version>
                </dependency>
                <dependency>
                  <groupId>org.apache.flink</groupId>
                    <artifactId>flink-ml_2.10</artifactId>
                    <version>${flink.version}</version>
                  </dependency>
        </dependencies>

        <profiles>
                <profile>
                        <!-- Profile for packaging correct JAR files -->
                        <id>build-jar</id>
                        <activation>
                                <activeByDefault>false</activeByDefault>
                        </activation>
                        <dependencies>
                                <dependency>
                                        <groupId>org.apache.flink</groupId>
                                        <artifactId>flink-java</artifactId>
                                        <version>${flink.version}</version>
                                        <scope>provided</scope>
                                </dependency>
                                <dependency>
                                        <groupId>org.apache.flink</groupId>
                                        <artifactId>flink-streaming-java_2.10</artifactId>
                                        <version>${flink.version}</version>
                                        <scope>provided</scope>
                                </dependency>
                                <dependency>
                                        <groupId>org.apache.flink</groupId>
                                        <artifactId>flink-clients_2.10</artifactId>
                                        <version>${flink.version}</version>
                                        <scope>provided</scope>
                                </dependency>
                        </dependencies>

                        <build>
                                <plugins>
                                        <!-- disable the exclusion rules -->
                                        <plugin>
                                                <groupId>org.apache.maven.plugins</groupId>
                                                <artifactId>maven-shade-plugin</artifactId>
                                                <version>2.4.1</version>
                                                <executions>
```

```
                                                        <execution>
                                                                <phase>package</phase>
                                                                <goals>
                                                                        <goal>shade</goal>
                                                                </goals>
                                                                <configuration>
                                                                        <artifactSet>
                                                                                <excludes
combine.self="override"></excludes>
                                                                        </artifactSet>
                                                                </configuration>
                                                        </execution>
                                                </executions>
                                        </plugin>
                                </plugins>
                        </build>
                </profile>
        </profiles>

        <build>
                <plugins>
                        <!-- We use the maven-shade plugin to create a fat jar that contains all dependencies
                        except flink and it's transitive dependencies. The resulting fat-jar can be executed
                        on a cluster. Change the value of Program-Class if your program entry point changes. -->
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-shade-plugin</artifactId>
                                <version>2.4.1</version>
                                <executions>
                                        <!-- Run shade goal on package phase -->
                                        <execution>
                                                <phase>package</phase>
                                                <goals>
                                                        <goal>shade</goal>
                                                </goals>
                                                <configuration>
                                                        <artifactSet>
                                                                <excludes>
                                                                        <!-- This list contains all dependencies of
flink-dist
                                                                        Everything else will be packaged into the
fat-jar
                                                                        -->
                                                                        <exclude>org.apache.flink:flink-
annotations</exclude>
                                                                        <exclude>org.apache.flink:flink-shaded-
hadoop1</exclude>
                                                                        <exclude>org.apache.flink:flink-shaded-
hadoop2</exclude>
                                                                        <exclude>org.apache.flink:flink-shaded-
curator-recipes</exclude>
                                                                        <exclude>org.apache.flink:flink-
core</exclude>
                                                                        <exclude>org.apache.flink:flink-
java</exclude>
                                                                        <exclude>org.apache.flink:flink-
scala_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-
runtime_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-
optimizer_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-
clients_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-
avro_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-examples-
batch_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-examples-
streaming_2.10</exclude>
                                                                        <exclude>org.apache.flink:flink-streaming-
```

```
java_2.10</exclude>

                                                                    <!-- Also exclude very big transitive
dependencies of Flink

                                                                    WARNING: You have to remove these
excludes if your code relies on other
                                                                    versions of these dependencies.

                                                                    -->
                                                                    <exclude>org.scala-lang:scala-
library</exclude>
                                                                    <exclude>org.scala-lang:scala-
compiler</exclude>
                                                                    <exclude>org.scala-lang:scala-
reflect</exclude>
                                                                    <exclude>com.amazonaws:aws-java-
sdk</exclude>
                                                                    <exclude>com.typesafe.akka:akka-
actor_*</exclude>
                                                                    <exclude>com.typesafe.akka:akka-
remote_*</exclude>
                                                                    <exclude>com.typesafe.akka:akka-
slf4j_*</exclude>
                                                                    <exclude>io.netty:netty-all</exclude>
                                                                    <exclude>io.netty:netty</exclude>
                                                                    <exclude>commons-fileupload:commons-
fileupload</exclude>
                                                                    <exclude>org.apache.avro:avro</exclude>
                                                                    <exclude>commons-collections:commons-
collections</exclude>
                                                                    <exclude>org.codehaus.jackson:jackson-
core-asl</exclude>
                                                                    <exclude>org.codehaus.jackson:jackson-
mapper-asl</exclude>
            <exclude>com.thoughtworks.paranamer:paranamer</exclude>
                                                                    <exclude>org.xerial.snappy:snappy-
java</exclude>
                                                                    <exclude>org.apache.commons:commons-
compress</exclude>
                                                                    <exclude>org.tukaani:xz</exclude>
            <exclude>com.esotericsoftware.kryo:kryo</exclude>
            <exclude>com.esotericsoftware.minlog:minlog</exclude>
            <exclude>org.objenesis:objenesis</exclude>
                                                                    <exclude>com.twitter:chill_*</exclude>
                                                                    <exclude>com.twitter:chill-java</exclude>
                                                                    <exclude>com.twitter:chill-
avro_*</exclude>
                                                                    <exclude>com.twitter:chill-
bijection_*</exclude>
                                                                    <exclude>com.twitter:bijection-
core_*</exclude>
                                                                    <exclude>com.twitter:bijection-
avro_*</exclude>
                                                                    <exclude>commons-lang:commons-
lang</exclude>
                                                                    <exclude>junit:junit</exclude>
                                                                    <exclude>de.javakaffee:kryo-
serializers</exclude>
                                                                    <exclude>joda-time:joda-time</exclude>
                                                                    <exclude>org.apache.commons:commons-
lang3</exclude>
                                                                    <exclude>org.slf4j:slf4j-api</exclude>
                                                                    <exclude>org.slf4j:slf4j-log4j12</exclude>
                                                                    <exclude>log4j:log4j</exclude>
                                                                    <exclude>org.apache.commons:commons-
```

```
math</exclude>

        <exclude>org.apache.sling:org.apache.sling.commons.json</exclude>
                                                        <exclude>commons-logging:commons-
logging</exclude>
                                                        <exclude>commons-codec:commons-
codec</exclude>

        <exclude>com.fasterxml.jackson.core:jackson-core</exclude>

        <exclude>com.fasterxml.jackson.core:jackson-databind</exclude>

        <exclude>com.fasterxml.jackson.core:jackson-annotations</exclude>
                                                        <exclude>stax:stax-api</exclude>
                                                        <exclude>com.typesafe:config</exclude>

        <exclude>org.uncommons.maths:uncommons-maths</exclude>

        <exclude>com.github.scopt:scopt_*</exclude>
                                                        <exclude>commons-io:commons-
io</exclude>
                                                        <exclude>commons-cli:commons-
cli</exclude>
                                                                </excludes>
                                                        </artifactSet>
                                                        <filters>
                                                                <filter>
                                                                        <artifact>org.apache.flink:*</artifact>
                                                                        <excludes>
                                                                                <!-- exclude shaded google but
include shaded curator -->

        <exclude>org/apache/flink/shaded/com/**</exclude>
                                                                                <exclude>web-
docs/**</exclude>
                                                                        </excludes>
                                                                </filter>
                                                                <filter>
                                                                        <!-- Do not copy the signatures in the
META-INF folder.
                                                                        Otherwise, this might cause
SecurityExceptions when using the JAR. -->
                                                                        <artifact>*:*</artifact>
                                                                        <excludes>
                                                                                <exclude>META-
INF/*.SF</exclude>
                                                                                <exclude>META-
INF/*.DSA</exclude>
                                                                                <exclude>META-
INF/*.RSA</exclude>
                                                                        </excludes>
                                                                </filter>
                                                        </filters>
                                                        <transformers>
                                                                <!-- add Main-Class to manifest file -->
                                                                <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                                                        <mainClass>letters.Job</mainClass>
                                                                </transformer>
                                                        </transformers>

        <createDependencyReducedPom>false</createDependencyReducedPom>
                                                </configuration>
                                        </execution>
                                </executions>
                        </plugin>

                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
```

```
                                        <version>3.1</version>
                                        <configuration>
                                                <source>1.7</source> <!-- If you want to use Java 8, change this to "1.8" -->
                                                <target>1.7</target> <!-- If you want to use Java 8, change this to "1.8" -->
                                        </configuration>
                                </plugin>
                        </plugins>
                </build>
                <groupId>org.apache.flink</groupId>
                <artifactId>flink-ml_2.10</artifactId>
</project>
```

Other useful comments:

HDFS connector

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-filesystem_2.10</artifactId>
  <version>1.1-SNAPSHOT</version>
</dependency>
```

Add this to pom.xml and then do not forget to refresh the pom.xml – following that the system will invoke the new artifact and set up import.

Start Hadoop (after stopping)
for x in `cd /etc/init.d ; ls hadoop-hdfs-*` ; do sudo service $x start ; done

Install Table: - use maven with the following line:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table_2.10</artifactId>
  <version>1.1-SNAPSHOT</version>
  </dependency>
```

Machine Learning Maven:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-ml_2.10</artifactId>
  <version>1.1-SNAPSHOT</version>
</dependency>
```