Technische Universität Berlin
Database Systems and Information Management Group
C. Boden, S. Schelter, J. Soto, and Prof. Dr. Volker Markl

Scalable Data Science: Systems and Methods
WS 2015-2016

Second Assignment

Classification, Recommender Systems, and Stream Processing

Due on December 18th at 12 noon

I. Classification (Total: 4 pts.)

In this exercise, you will familiarise yourself with classification.

A. Naive Bayes

Implement a Naive Bayes classifier using Apache Flink. Evaluate your implementation on a data set of postings from 20 newsgroups (check out `http://qwone.com/~jason/20Newsgroups/` for a description of the data). Clone the classification source code from the Git Lab assignment repository and have a look at *de.tuberlin.dima.aim3.assignment2*. Be sure to adjust *de.tuberlin.dima.aim3.assignment2.Config* to point to the location of your local project. You will have to complete the three provided classes *Training*, *Classification* and *Evaluator*.

1. In *Training.java* you have to complete the two flink data flows that should compute the conditional counts of words per label (<label, word, count>) as well as the summed counts of all word counts per label (<label, counts>).
2. In *Classification.java* you have to implement the classification function.
3. In *Evaluator.java* you have to complete a function that computes the accuracy of your classifier on a provided test data set. Your classifier should produce an accuracy of well over 80%. If not you maybe forgot to introduce a smoothing parameter in your classifier.

We have also provided a test set without labels (*secrettest.dat*). Run your trained classifier and write the output to a file. It should contain the number, label, and probability assigned to each data point in a tab separated manner (e.g., [1]<tab>talk.politics.guns<tab>0.123). Please also upload this file together with your patch in ISIS.

B. Background Notes

1. **Naive Bayes Classifier: Theory**
   As discussed in class, a naive Bayes classifier models a joint distribution over a label Y and a set of observed random variables or features, $(F_1, F_2 \dots F_n)$ using the assumption that the full joint distribution can be factored as follows (features are assumed to be conditionally independent given the label):

$$P(F_1, F_2 \dots F_n) = P(Y) \prod_i P(F_i|Y)$$

To classify a datum, we can find the most probable label given the feature values for each pixel, using Bayes theorem:

$$P(y|f_1, \ldots, f_m) = \frac{P(f_1, \ldots, f_m|y)P(y)}{P(f_1, \ldots, f_m)} = \frac{P(y)\prod_{i=1}^{m} P(f_i|y)}{P(f_1, \ldots, f_m)}$$

$$argmax_y P(y|f_1 \ldots f_m) = argmax_y \frac{P(y)\prod_{i=1}^{m} P(f_i|y)}{P(f_1, \ldots, f_m)} = argmax_y P(y)\prod_{i=1}^{m} P(f_i|y)$$

Given that multiplying many probabilities can result in underflow, instead we will compute log probabilities which have the same argmax:

$$argmax_y \log P(y|f_1 \ldots f_m) = argmax_y \left\{ \log P(y) + \sum_{i=1}^{m} \log P(f_i|y) \right\}$$

To compute logarithms, use `Math.log()` function. Put simply, you will have to compute the most likely label given the data (text in the posting to be classified). This means you have to compute the probability for each label and then return the label which recieves the highest probability from your model for the document in question.

2. **Parameter Estimation**
   Our naive Bayes model has several parameters to estimate. One parameter is the prior distribution over labels (the names of the newsgroups), $P(Y)$. for simplicity we assume a uniform prior accross all classes. The term $\log P(y)$ can thus be ignored in the argmax computation. The other parameters to estimate are the conditional probabilities of our features given each label y: $P(F_i|Y = y)$, where is the conditional probability of a word w occurring in a posting of class(topic) y. We do this for each possible feature value (each word in the vocabulary).

$$\hat{P}(w|Y = y) = \frac{count(w, y)}{\sum_{w'} count(w', y)}$$

   where $count(w, y)$ is the number of times word w occured in the training examples of label y.

3. **Smoothing**
   Your current parameter estimates are unsmoothed, that is, you are using the empirical estimates for the parameters. These estimates are rarely adequate in real systems. Minimally, we need to make sure that no parameter ever receives an estimate of zero, but good smoothing can boost accuracy quite a bit by reducing overfitting. In this assignment, we use Laplace smoothing, which adds k counts to every possible observation value:

$$P(w|Y = y) = \frac{count(w, y) + k}{\sum_{w'} (count(w', y) + k)}$$

   If $k = 0$, the probabilities are unsmoothed. As k grows larger, the probabilities are smoothed more and more.

II. Recommender Systems: Item-based Collaborative Filtering (Total 2 pts.)

Suppose you are given a dataset of ratings as depicted in the table below:

	Movie A	Movie B	Movie C	Movie D
Ryan	1	5	3	5
Stavros	-	-	2	-
Brahma	-	4	1	1
Brodie	4	3	-	2
Zosimus	-	5	1	-

1. Compute the **item similarity matrix** (denoted by R) using **Pearson Correlation** as a measure.
2. Compute the **item similarity matrix** (denoted by S) using **Jaccard Coefficient** as a measure.
3. Use matrices R and S to compute the corresponding ratings that *Zosimus* would give to *Movies A & D*.
4. Use matrices R and S to compute the corresponding ratings that *Stavros* would give to *Movies A & B*.

For each sub-problem 1-4, show how you arrived at your solution.

III. Stream Processing (Total: 4 pts.)

In this exercise, you will familiarise yourself with streaming in Flink.

A. Preparation

As a starting point, read both the Flink Programming Guide[1] and the Flink Streaming Manual[2]. In addition, for this specific exercise, you should use Apache Flink version 0.10.1.

B. Foundations

Prior to answering the following questions, compare the Word Count example referenced in the Batch API[3] with the corresponding one referenced in the Streaming API[4].

1. What are the key differences between batch and stream data sources?
2. What problem will we encounter, if blocking operators, such as aggregations are used in streaming queries?
3. How do streaming engines address the problem referenced just above? Hint: Examine the aforementioned Word Count examples.
4. Batch processing engines like Apache Spark simulate streaming behavior using discretized streams (D-Streams). What is the main idea behind D-Streams?

[1] Programming Guide - `https://ci.apache.org/projects/flink/flink-docs-release-0.10/apis/programming_guide.html`

[2] Streaming API - `https://ci.apache.org/projects/flink/flink-docs-release-0.10/apis/streaming_guide.html`

[3] Word Count Batch - `https://github.com/apache/flink/blob/master/flink-examples/flink-java-examples/src/main/java/org/apache/flink/examples/java/wordcount/WordCount.java`

[4] Word Count Stream - `https://github.com/apache/flink/blob/master/flink-streaming-examples/src/main/java/org/apache/flink/streaming/examples/windowing/WindowWordCount.java`

5. What are the disadvantages attributed to using D-Streams on a batch processing engine, as opposed to using a runtime engine that offers native streaming support?

C. Monitoring a Fleet of Taxis

Suppose an Administrator (named Susan) at a Taxi Corporate Office monitors her companies fleet using a software system that tracks GPS signals. She has contacted you and requests an upgrade to the software system, i.e., extend its capabilities to monitor additional events described further below). Given the TopSpeedWindowing Example (see GitLab) as a starting point, adjust it to alert users (e.g., via print/display) to particular events.

1. Examine the TopSpeedWindowing example and illustrate the query in a flow graph. Describe which data are transferred along the different edges.
2. Adjust the (TopSpeedWindowing specific) stream data generator as follows:
   (a) Urban taxi drivers seldom drive faster than 50 km/h, but do so on occasion.
   (b) Urban taxi drivers experience long wait times between customers periodically.
3. Using your adapted stream data generator, display an alert, whenever a taxi drives faster than 50 km/h.
4. Using your adapted stream data generator, display an alert that continuously notifies (each second) about all taxis that are idle for over 5 seconds.
5. Update your illustration to reflect all of the changes you have made.

Deadline and General Instructions

Please submit your results to ISIS in a zip archive with the following structure and naming conventions until the above specified deadline:

```
aim3-WS1516-<name>.zip
├── author.txt (contains your name and matriculation number)
├── task I
│   └── the patch files to be submitted
├── task III
│   └── the patch files to be submitted
└── documentation.pdf (answers to the questions posed in each of the exercises)
```