Increment 3

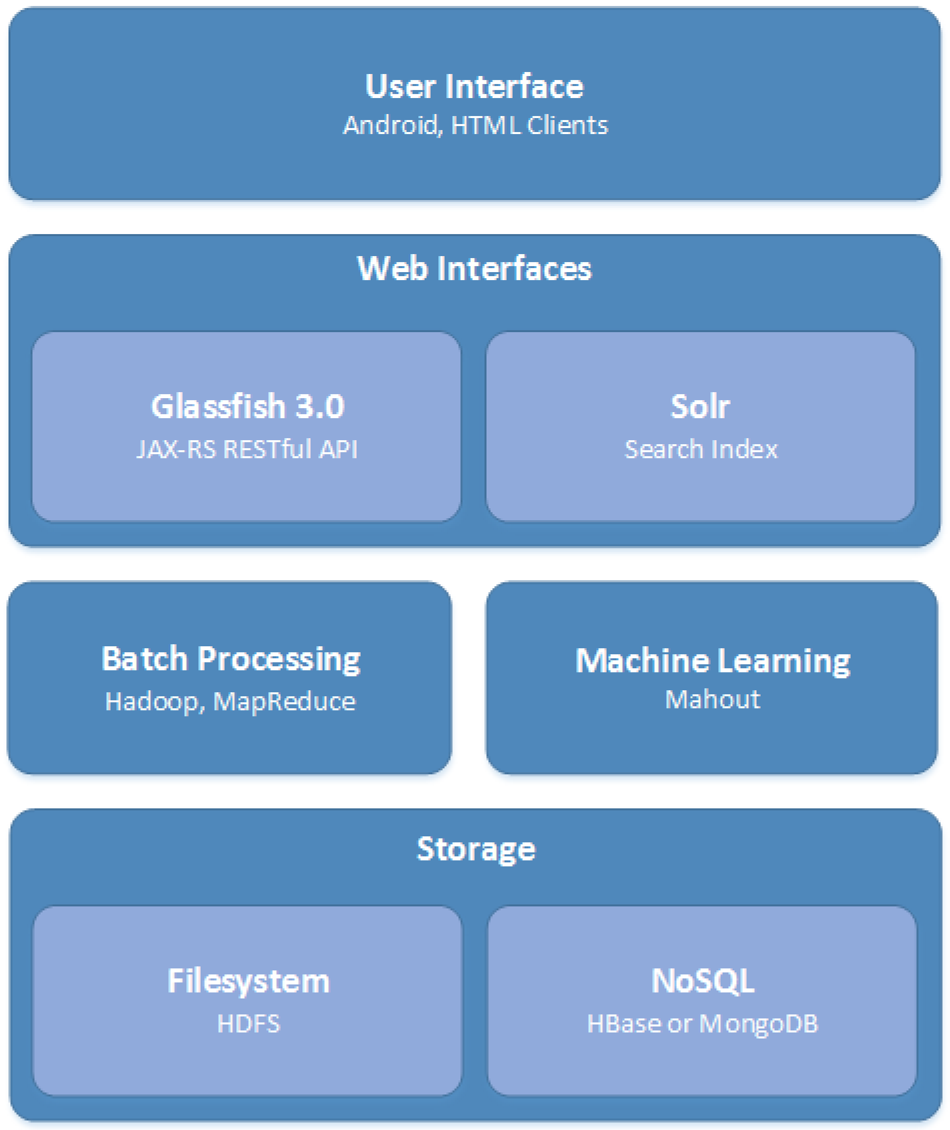
Group 6

James Clark
Tirumala Konireddy
Jagadish Tirumalasetty
Wang Zhang

University of Missouri - Kansas City

CS 560: Knowledge Discovery and Management
Dr. Yugyung Lee
April 18, 2014

System Architecture Diagram



Domain Model

Data Sources

The following four agencies contribute to a single restful API web service, providing summary data as it pertains to safety and recall information.

- Food and Drug Administration (FDA)
- U.S. Department of Agriculture's Food Safety and Inspection Service (UDSA FSIS)
- National Highway Traffic Safety Administration (NHTSA)
- Consumer Product Safety Commission (CPSC)

The web service features are outlined at http://search.digitalgov.gov/developer/recalls.html. In general, the service provides access to all recalls since 1966 but requests are limited to 50 results per page. A general query http://api.usa.gov/recalls/search.json? without parameters reveals ~26,000 data records at the time of this writing and it continues to grow daily. The challenge is to collect all data items under the specific criteria.

Every request lists the total matching records, but displays a maximum of 50 items per request. It is possible to page forward a maximum of 20 times. The result is a (50x20) 1000 record constraint. While it is possible to send a query that returns 26,000 results, it will never be possible to download all the results in a single request to the web server. Working inside these constraints is possible. Here's how:

Fetching data for this project, a recursive algorithm was used to collect results. Working down from 2014, the program ran out of memory at 1994. The algorithm could be restarted from that time until it runs out of memory again, but it makes more sense to rewrite it without recursion.

After running this data collection process, a text file is produced with over 800 lines of JSON strings each including up to 50 records, resulting in a data set that is both wide and tall albeit at relatively small scale. In all, the file is 52.5 MB in size. Further processing is required to reduce the data to consumable formats for Mahout, Weka, or R.

Discovery

The data has more depth. It's possible to follow the data to new locations to learn even more information. For example, one USDA record includes a web url to a page on a different server where a more granular view includes such additional information as volumes of shipments of the product in question. The link discusses a recent salmonella contamination. This leads to a lot of related Twitter discussion, so there is a big social data presence related to this one data record.

Methodologies

Front End Methodology

1. Get User credentials from user.
2. Send a query about the requested alerts through webservice to data-base server.
3. Retrieve the whole json array of alerts using hadoop file system that were stored in the server under that user credentials.
4. Display those alerts on the user home screen and now user can see all his alerts.
5. If user request any new category alerts through search, the app sends a query of 'search text' to server through Solr to analyze search text and then hadoop file system to is used to search the category in the server to retrieve that subscription array.
6. If user wants to subscribe that service, then that service is appended to the user's json array in server using mahout and from then the that service alerts were synced with the recent recall alerts that were shown in home screen.

Back End Methodology

1. Our system keeps on updating the rss feeds and websites data document(google document) every day for any appended data to them.
2. If any new data or information is detected, our system stores that data into a temporary dynamic variable.
3. Run adaptive decision tree using mahout where we use Case base reasoning and naïve bayes classifier to check whether there is any relevant information to alert or not.
4. If the decision tree returns a alert of some category, store that alert in server of json array.
5. We are using a naive Bayes classifier while continuing research in new algorithms and systems that improve relevance.

Algorithms

Analytic Tools

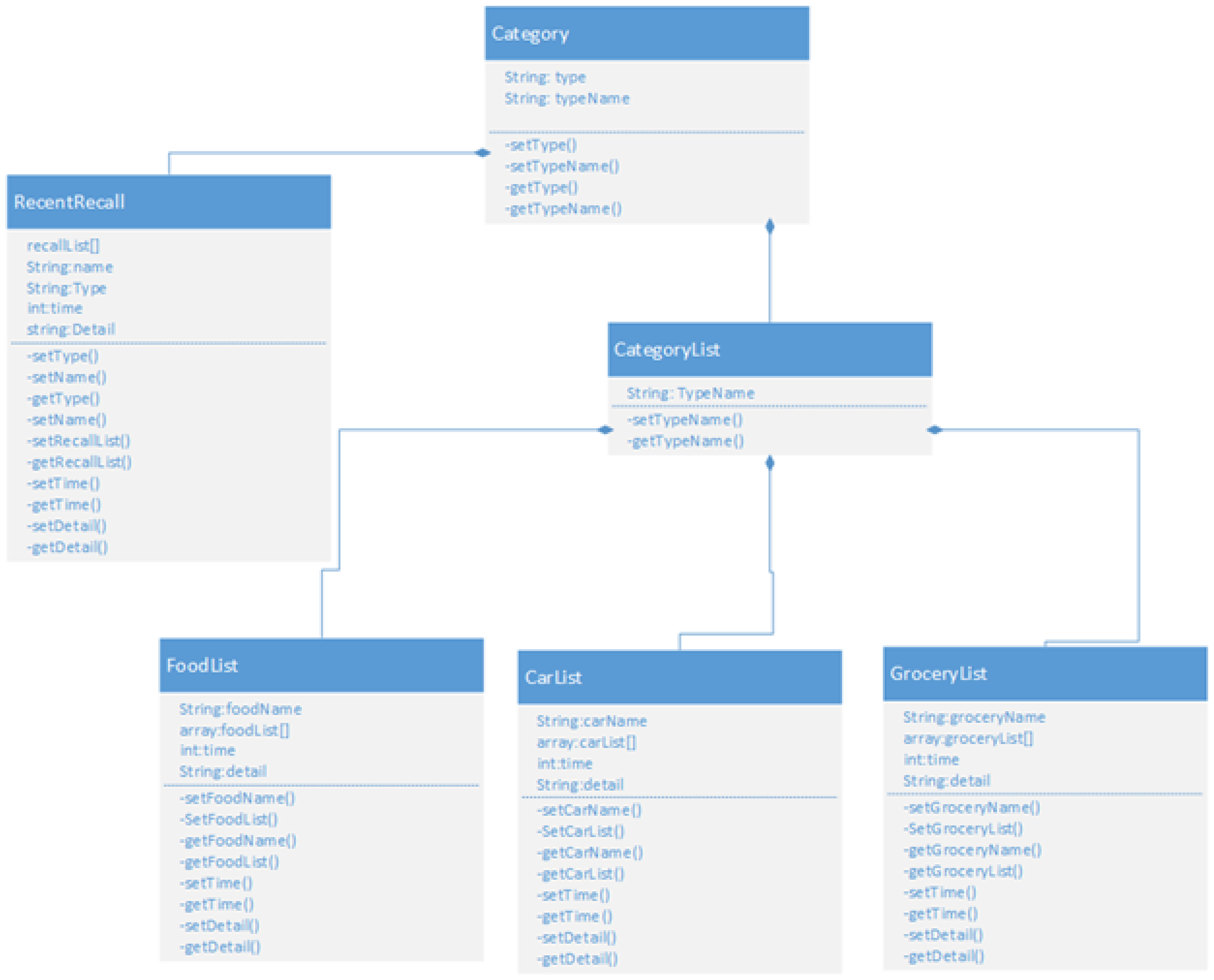
1. Mahout: For implementing Adaptive decision tree, Case base reasoning and naïve bayes classifier.
2. Hadoop: For using distributed system to run Mapping, reducing functions and store data into server(json table).
3. Solr: To map user entered search text to relevant category while searching and retrieving data from server. We may use it for analyzing data from rss feeds and html files.

Analytical Tasks

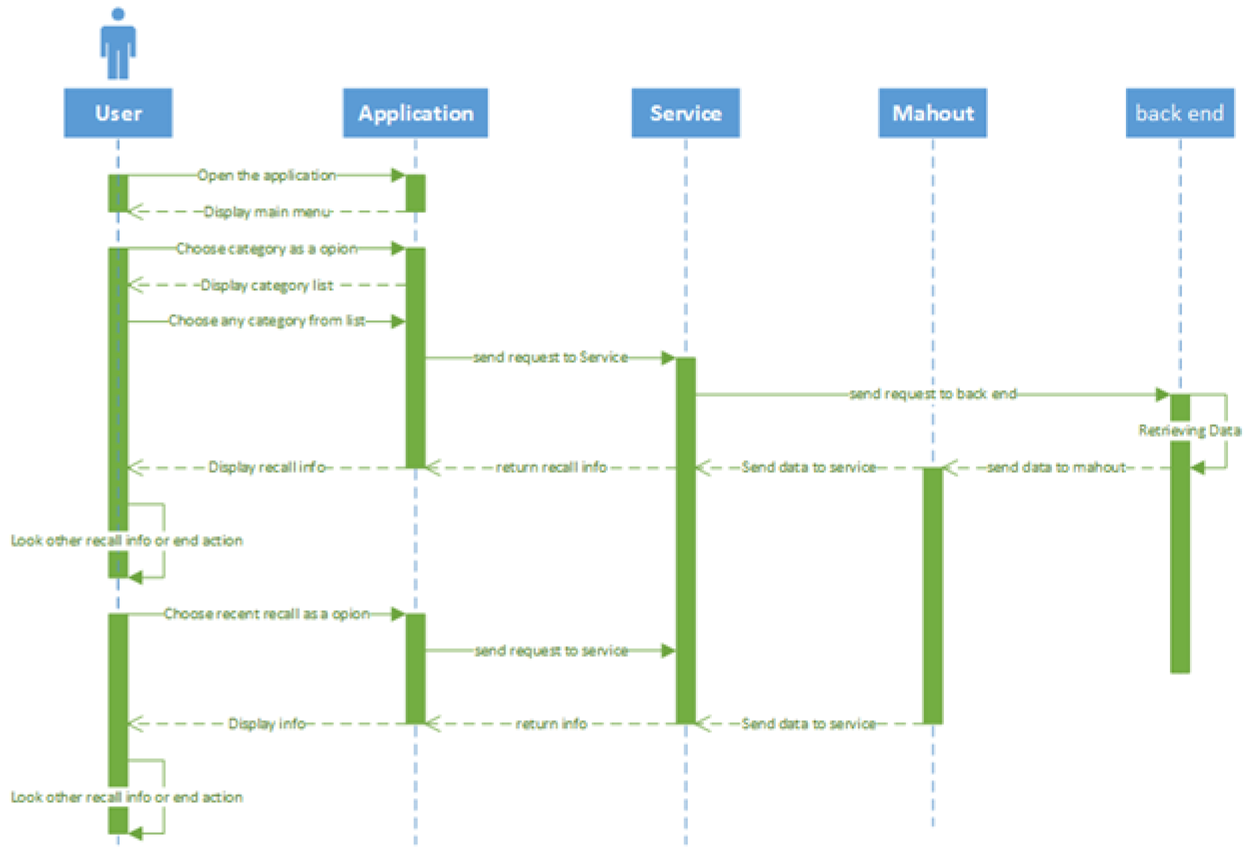
1. Collection of data : We get data from http://www-odi.nhtsa.dot.gov/downloads/flatfiles.cfm about all the product recalls that had done in past till now. Now, we are manually monitor and download the updated recall data flat .txt file.

Application Specification

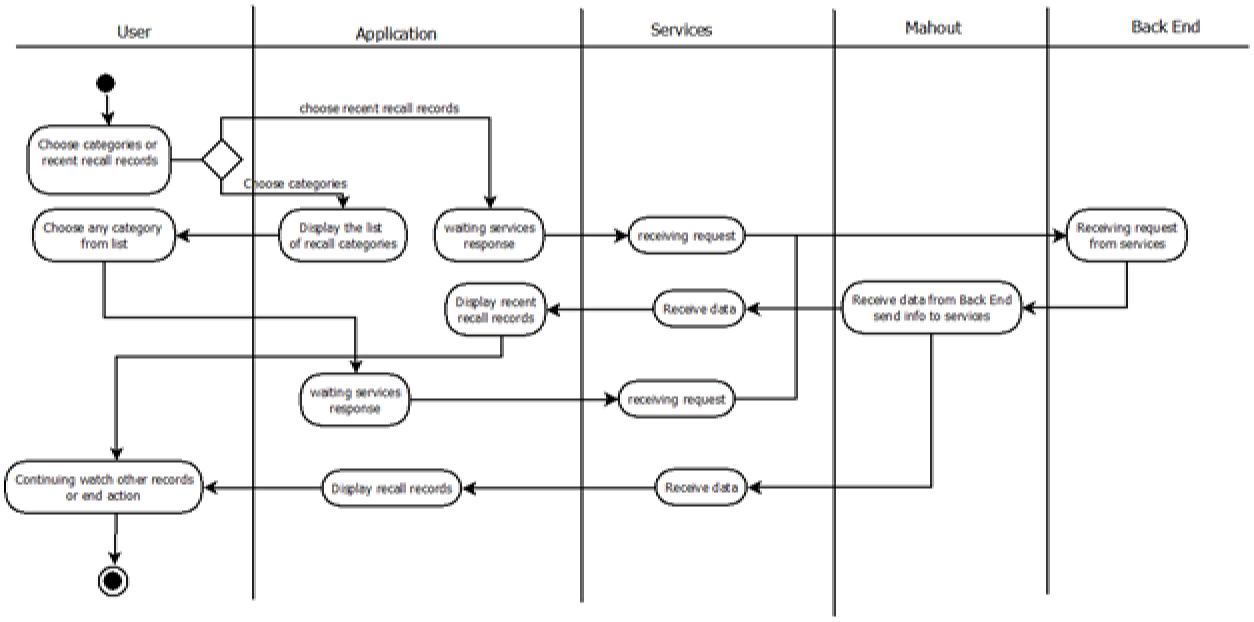
Class diagram



Sequence diagram



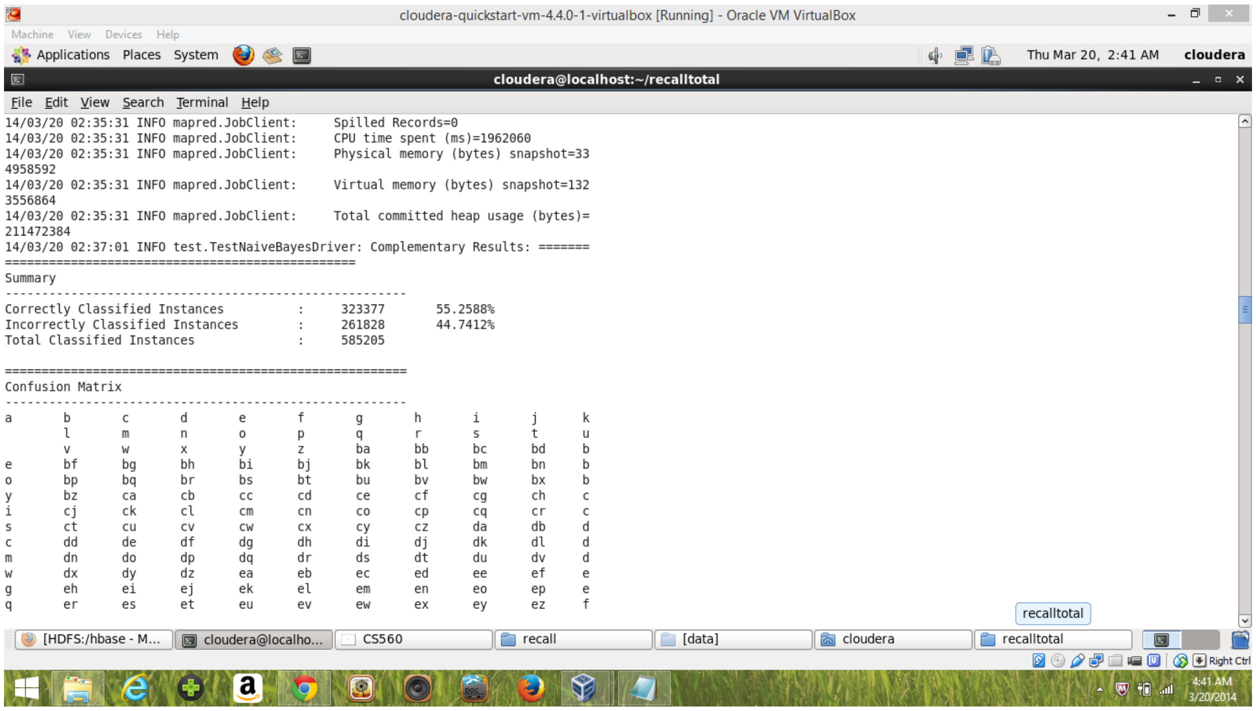
Activity Diagram (workflow, data, task)


User
Application
Services
Mahout
Back End
choose recent recall records
Choose categories or recent recall records
Choose categories
Choose any category from list
Display the list of recall categories
waiting services response
receiving request
Receiving request from services
Display recent recall records
Receive data
Receive data from Back End send info to services
waiting services response
receiving request
Continuing watch other records or end action
Display recall records
Receive data

Implementation

Classification

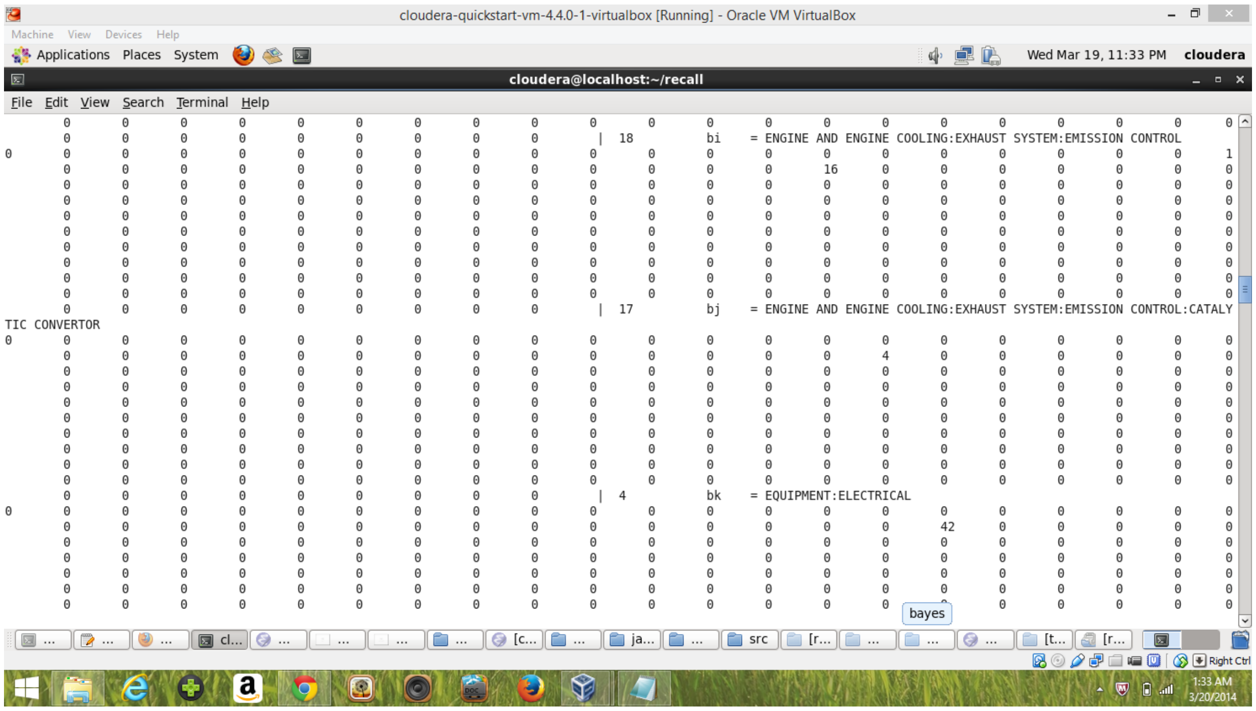
We are using Naive bayes classification method to classify the recalls of unknown category in list to find the category of the recall present in raw data. We got very good accuracy of about 90.27% when we had trained the classifier with 10,040 data entries.



cloudera-quickstart-vm-4.4.0-1-virtualbox [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places System
Thu Mar 20, 2:41 AM
cloudera
cloudera@localhost:~/recalltotal
File Edit View Search Terminal Help
recalltotal
[HDFS:/hbase - M...
cloudera@localho...
CS560
recall
[data]
cloudera
recalltotal
Right Ctrl
4:41 AM
3/20/2014

```
14/03/20 02:35:31 INFO mapred.JobClient:     Spilled Records=0
14/03/20 02:35:31 INFO mapred.JobClient:     CPU time spent (ms)=1962060
14/03/20 02:35:31 INFO mapred.JobClient:     Physical memory (bytes) snapshot=33
4958592
14/03/20 02:35:31 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=132
3556864
14/03/20 02:35:31 INFO mapred.JobClient:     Total committed heap usage (bytes)=
211472384
14/03/20 02:37:01 INFO test.TestNaiveBayesDriver: Complementary Results: =======
================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :     323377       55.2588%
Incorrectly Classified Instances        :     261828       44.7412%
Total Classified Instances              :     585205

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       h       i       j       k
        l       m       n       o       p       q       r       s       t       u
        v       w       x       y       z       ba      bb      bc      bd      b
e       bf      bg      bh      bi      bj      bk      bl      bm      bn      b
o       bp      bq      br      bs      bt      bu      bv      bw      bx      b
y       bz      ca      cb      cc      cd      ce      cf      cg      ch      c
i       cj      ck      cl      cm      cn      co      cp      cq      cr      c
s       ct      cu      cv      cw      cx      cy      cz      da      db      d
c       dd      de      df      dg      dh      di      dj      dk      dl      d
m       dn      do      dp      dq      dr      ds      dt      du      dv      d
w       dx      dy      dz      ea      eb      ec      ed      ee      ef      e
g       eh      ei      ej      ek      el      em      en      eo      ep      e
q       er      es      et      eu      ev      ew      ex      ey      ez      f
```

Whereas, we got low accuracy of about 55.25% in classifying recall categories, when we used to train 1,000,000 data records as a result of lot of distortion in the training data. But, we get good results in classification of unknown recall message with 1 million data entries classifier than the 10,040 data entries trained. We had tried to increase the accuracy of classifier by some improvements in program, but it doesn't work. As, 1 million data entries trained classifier producing good results, we are using that classifier for classifying unknown recall categories.


cloudera-quickstart-vm-4.4.0-1-virtualbox [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places System
Wed Mar 19, 11:33 PM cloudera
cloudera@localhost:~/recall
File Edit View Search Terminal Help
bayes
Right Ctrl
1:33 AM
3/20/2014

```
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0        |  18          bi     = ENGINE AND ENGINE COOLING:EXHAUST SYSTEM:EMISSION CONTROL
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       1
        0       0       0       0       0       0       0       0       0       0       0       0       0       16      0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0        |  17          bj     = ENGINE AND ENGINE COOLING:EXHAUST SYSTEM:EMISSION CONTROL:CATALY
TIC CONVERTOR
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       4       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0        |  4           bk     = EQUIPMENT:ELECTRICAL
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       42      0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
```

Recommendation Algorithm

I had developed a new algorithm to work efficiently for our data by considering the likeliness of both interest and disinterest on a particular category with present subscriptions of user based on previous user’s subscriptions.
First, I am collecting all the categories present in the previous user’s subscriptions.
Then, I am calculating the number of occurrences of every item in the whole user’s data base. This number itself denotes the importance or priority of a particular item in recommending it.
Next, I am providing a index for all individual items with all other items as below.

Index for a recommending item B to user who subscribed to A = (sum of number of occurrences of B along with A in previous user’s subscriptions list) - (sum of number of disappearances of B when A is present in previous user’s subscriptions list)

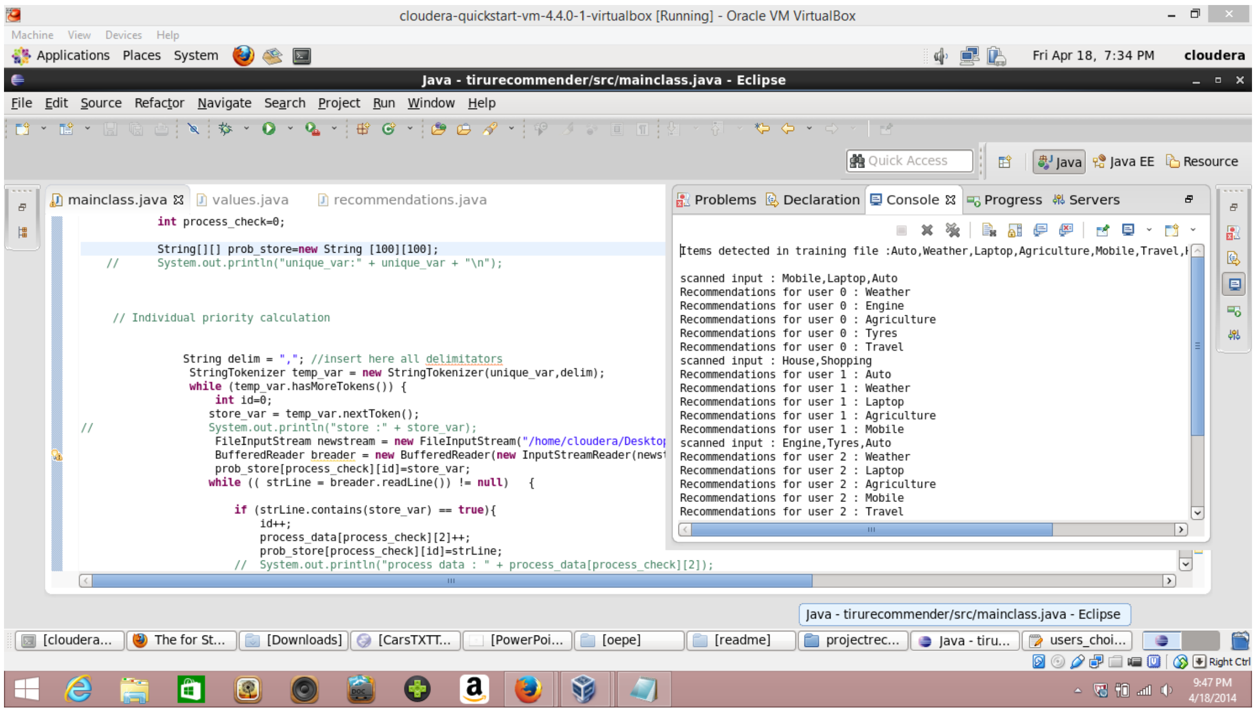
Then, We recommend a particular category in the entire list to a user subscribed for A and C which has the highest sum of indexes in A and C other than A and C.

Index of recommending B for user subscribed to A and C categories is
Index B (A and C)=Index B(A) + Index B(C)
Similarly, If i had A,B,C,D,E categories in my data base, then
Index D (A and C)=Index D(A) + Index D(C)
Index E (A and C)=Index E(A) + Index E(C)

Then, I recommend the category in B, D, E which has highest Index.

Using this algorithm, we can eliminate the unlikely combinations in providing recommendations.
For Example :
In amazon.com
if, user 1 had bought a Sony Laptop and a back pack bag
user 2 had bought a Samsung Laptop and a back pack bag too
Here, recommending both Sony Laptop and Samsung Laptop is somewhat good for user who bought a back pack bag.
Recommender algorithms which uses only the appearance of a object with the common object regardless the object type. This will provide a recommendation of a Samsung laptop to a user who had bought a Sony Laptop as it is bought by a user who had bought a back pack bag.
But, it is useless to recommend a Samsung Laptop for a user immediately after buying a Sony Laptop as it is atleast 2 years investment. This type of recommendations can be avoided by calculating the number disappearances along with the number of appearances when a particular item is present.

I had developed a program in java, to implement the above algorithm in providing recommendations to user.


cloudera-quickstart-vm-4.4.0-1-virtualbox [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places System
Fri Apr 18, 7:34 PM cloudera
Java - tirurecommender/src/mainclass.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java Java EE Resource
mainclass.java values.java recommendations.java
Problems Declaration Console Progress Servers

```
            int process_check=0;

            String[][] prob_store=new String [100][100];
  //        System.out.println("unique_var:" + unique_var + "\n");



   // Individual priority calculation


                String delim = ","; //insert here all delimitators
                 StringTokenizer temp_var = new StringTokenizer(unique_var,delim);
                 while (temp_var.hasMoreTokens()) {
                     int id=0;
                   store_var = temp_var.nextToken();
//                 System.out.println("store :" + store_var);
                    FileInputStream newstream = new FileInputStream("/home/cloudera/Desktop
                    BufferedReader breader = new BufferedReader(new InputStreamReader(newst
                    prob_store[process_check][id]=store_var;
                  while (( strLine = breader.readLine()) != null)   {

                        if (strLine.contains(store_var) == true){
                            id++;
                            process_data[process_check][2]++;
                            prob_store[process_check][id]=strLine;
                        //  System.out.println("process data : " + process_data[process_check][2]);
```

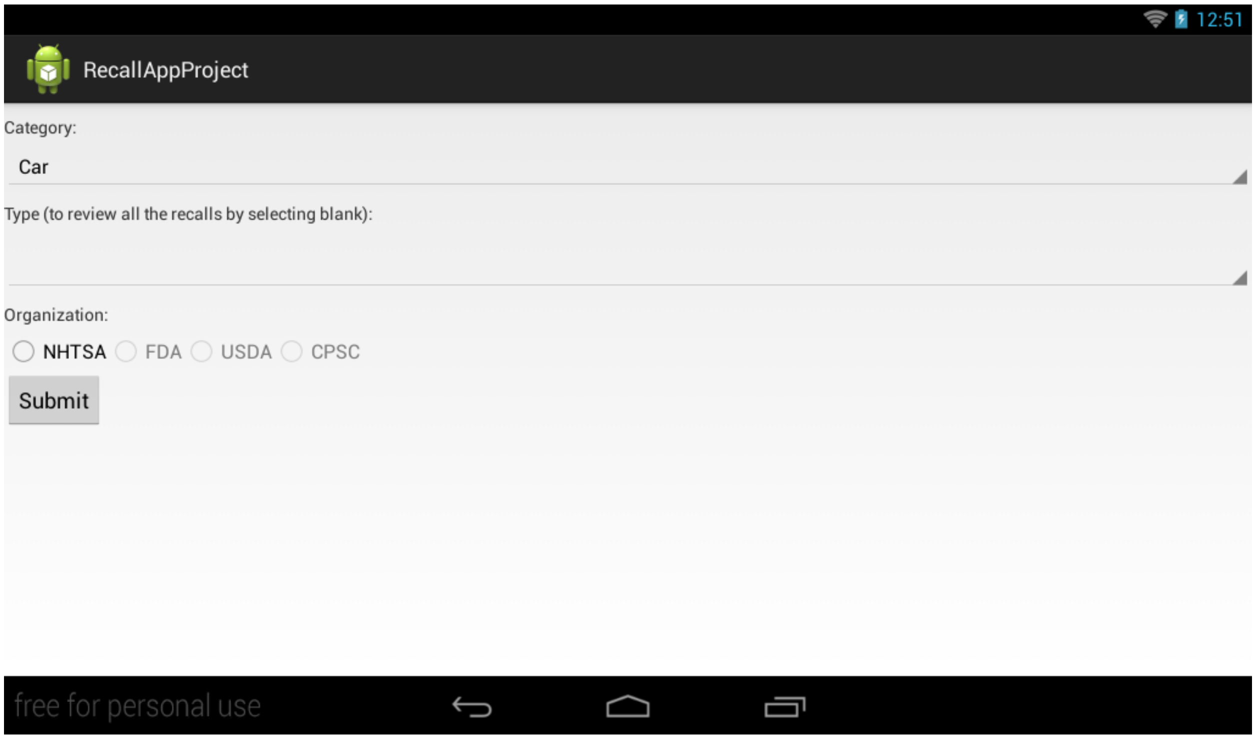
```
Items detected in training file :Auto,Weather,Laptop,Agriculture,Mobile,Travel,H

scanned input : Mobile,Laptop,Auto
Recommendations for user 0 : Weather
Recommendations for user 0 : Engine
Recommendations for user 0 : Agriculture
Recommendations for user 0 : Tyres
Recommendations for user 0 : Travel
scanned input : House,Shopping
Recommendations for user 1 : Auto
Recommendations for user 1 : Weather
Recommendations for user 1 : Laptop
Recommendations for user 1 : Agriculture
Recommendations for user 1 : Mobile
scanned input : Engine,Tyres,Auto
Recommendations for user 2 : Weather
Recommendations for user 2 : Laptop
Recommendations for user 2 : Agriculture
Recommendations for user 2 : Mobile
Recommendations for user 2 : Travel
```

Java - tirurecommender/src/mainclass.java - Eclipse
[cloudera... The for St... [Downloads] [CarsTXTT... [PowerPoi... [oepe] [readme] projectrec... Java - tiru... users_choi...
Right Ctrl
9:47 PM
4/18/2014

Implementation of User Interface

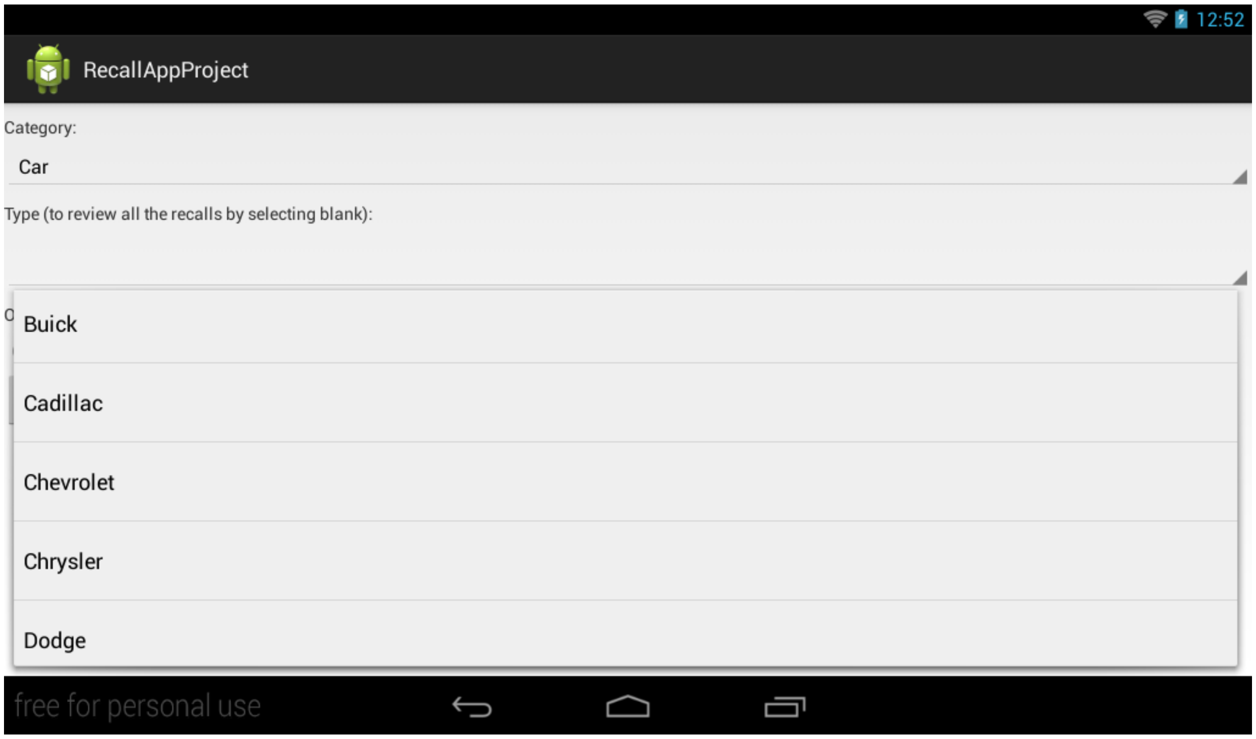
restructure the Android user interface to make app more acceptable and convenience to the user. for the new interface, we made two spinner. One is recall category, in this category spinner, user are able to choose four type of recalls, it includes car, food, drug and others.

another spinner is depended on the first spinner. if the user choose car in the first spinner, in the second spinner, user will be able to choose manufactory of the car. also user will be able to choose the organization who issued the recall. eventually, base on what user choose, the recall date will display to the user.



Category:
Car
Type (to review all the recalls by selecting blank):
Organization:
NHTSA FDA USDA CPSC

right now, this is user selection page. Through this page, user will be able to choose category, type and organization of recall. For organization, there are four organization issue recalls. NHTSA (National Highway Traffic Safety Administration) is organization that only issue recalls about car. FDA (Food and Drug Administration) issues food and drug recalls. USDA (U.S. Department of Agriculture's Food safety) issues food recalls. CPSC (Consumer Product Safety Commission) issues all kind of product. So for car category, we only enable user to select NHTSA, For drug category, we only enable FDA option for the user. For food category, we enable FDA and USDA options for the user. For others category, we enable CPSC for the users.



Category:

Car

Type (to review all the recalls by selecting blank):

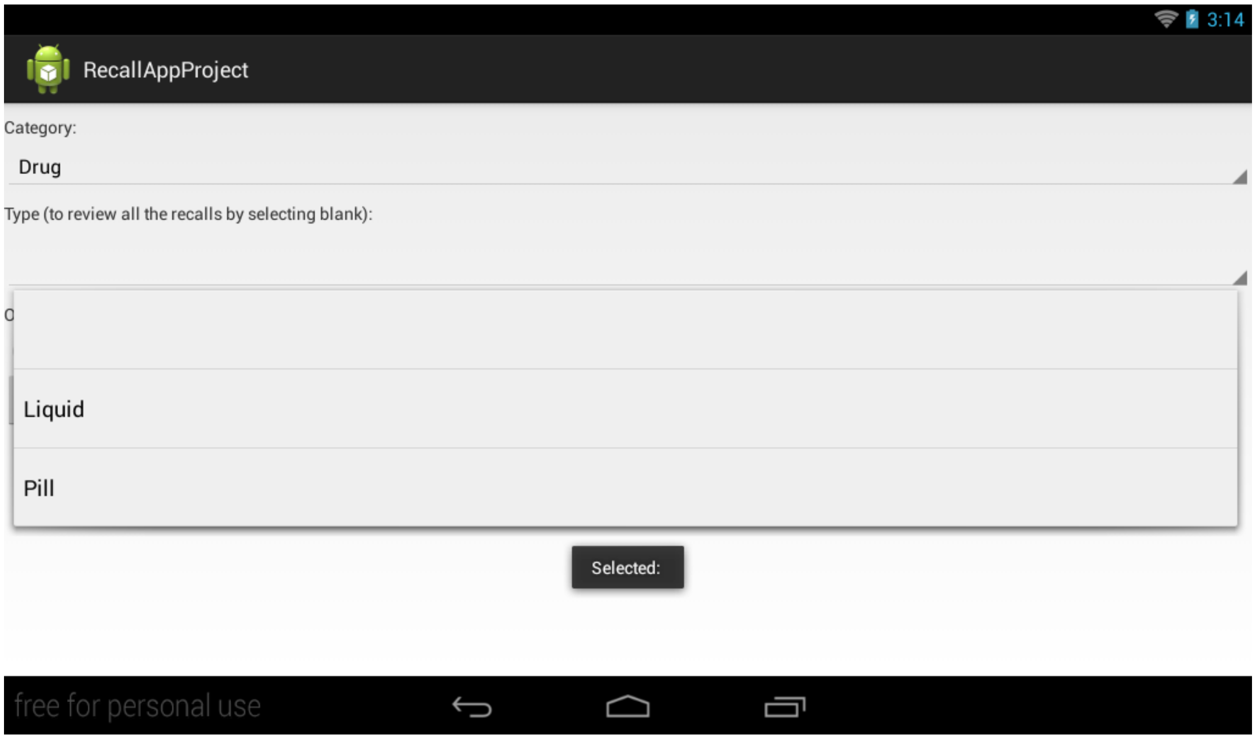
Buick

Cadillac

Chevrolet

Chrysler

Dodge



Category:

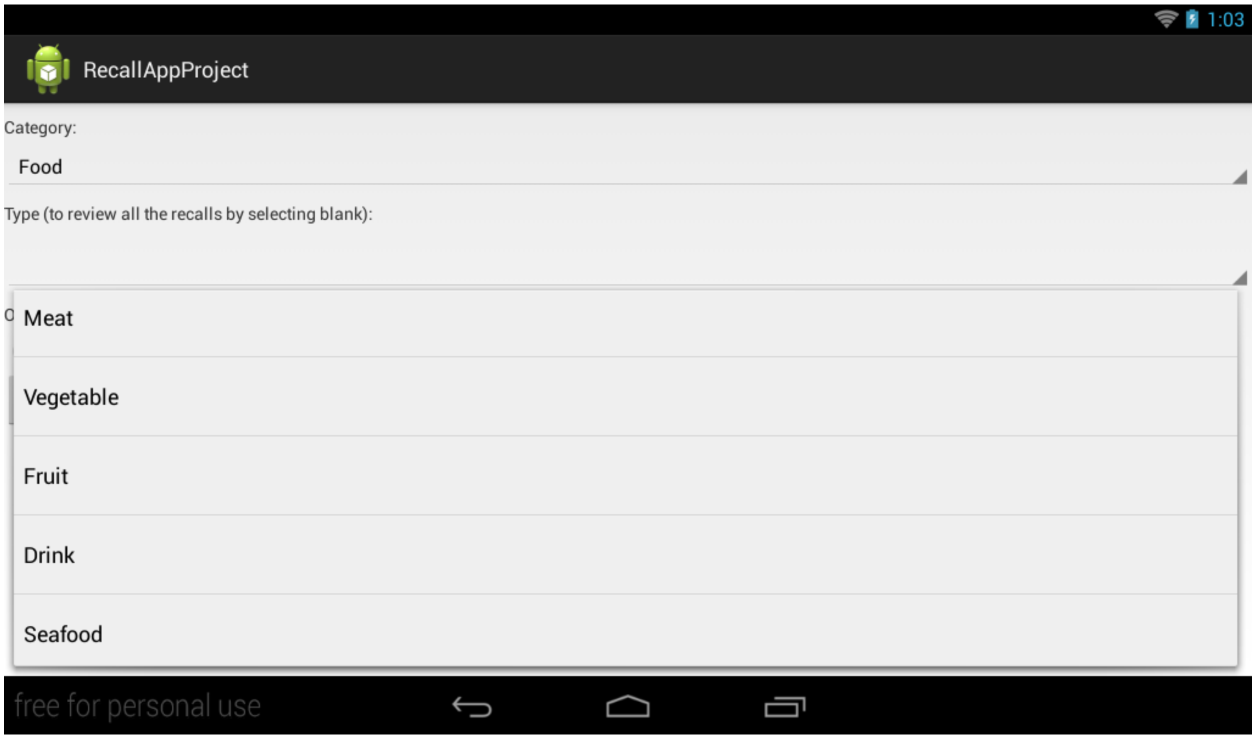
Drug

Type (to review all the recalls by selecting blank):

Liquid

Pill

Selected:



1:03

RecallAppProject

Category:

Food

Type (to review all the recalls by selecting blank):

Meat

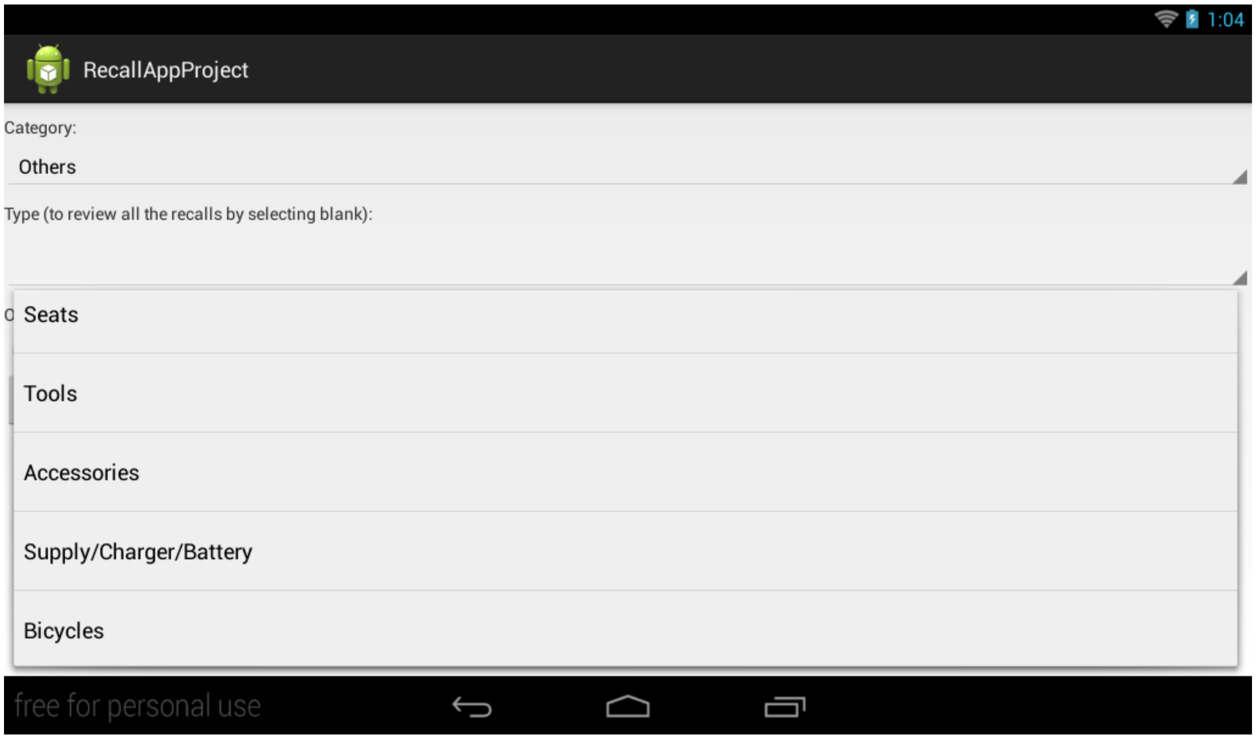
Vegetable

Fruit

Drink

Seafood

free for personal use



1:04

RecallAppProject

Category:

Others

Type (to review all the recalls by selecting blank):

Seats

Tools

Accessories

Supply/Charger/Battery

Bicycles

free for personal use

For different categories. it has different types.

Service Specification

We are hosting a limited REST api on the UMKC Glassfish server. The URL is http://134.193.136.127:8080/six.group.kdm/. It features search and update methods for Solr.

Project Timelines, Members, Task Responsibility

Implementation status report

Work completed

Wang developed client interfaces that pull data directly from usa.gov
Tiru wrote successful recommendation methods for Mahout
James developed GET and POST restful methods and hosted a JSP page summarizing them on UMKC's Glassfish server

Work to be completed

Full integration of machine learning to client through Solr index.
Project management on ScrumDo

Developing for Glassfish

Eclipse EE Indigo x86
Java JDK 6u32 x86
Glassfish 3.0 Open Source Edition

Production Servers

Solr 4.4
Glassfish 3.0 on JDK 6u32

Project Libraries

A list of external Java libraries included in this project, their versions, and their location

Apache Commons IO 2.6 - http://commons.apache.org/proper/commons-io/
Apache Commons Logging 1.1.3 - http://commons.apache.org/proper/commons-logging/
Apache HTTP Components 4.3.3 - https://hc.apache.org/
Simple Logging Facade for Java 1.7.7 - http://www.slf4j.org/
Solrj 4.7.1 - http://search.maven.org/#search|ga|1|solrj
JSONObject - https://github.com/douglascrockford/JSON-java
Joda-Time - http://www.joda.org/joda-time/