Non-Personalized Car Recommender System

Group:

- Mark Pit
- Toan Quach
- Hiep Nguyen

Instructor: Prof. Shafqat Ali Shad

21. Oct 2015

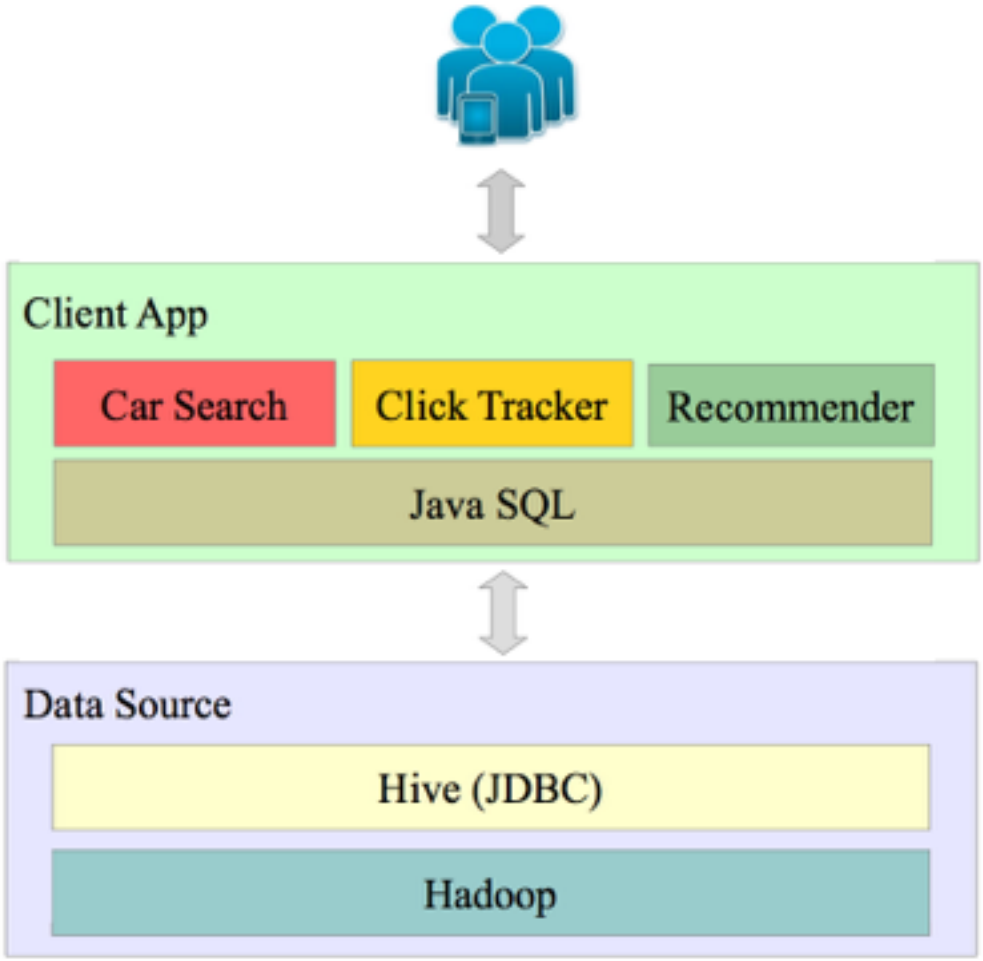
Agenda

- Overview
- Architecture
- Apriori Algorithm
- Recommendation Criteria
- Tools & Technologies
- Limitations
- Demo

Overview

- The domain is non-personalized, which means it gets data from other users to form a recommendation. This is done through the detection and storage of user clicks and searches and transforming it into a set of frequently accessed items using a data mining algorithm called Apriori.
- Datasets are scraped or crawled from data sources and pre-processed before importing to the HDFS using Hive. Data is then retrieved by the application through JDBC.

Architecture



Apriori Algorithm I

- Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.
- The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database.
- Example
  - Assume a large set of car ID's taken from the database including the target user's clicks and search history:

Itemsets
{1,2,3,4}
{1,2,4}
{1,2}
{2,3,4}
{2,3}
{3,4}
{2,4}

Apriori Algorithm II

- We will use Apriori to determine the frequent item sets of this database. To do so, we will say that an item set is frequent if it appears in at least 3 transactions of the database: the value 3 is the support threshold.
- The first step of Apriori is to count up the number of occurrences, called the support, of each member item separately, by scanning the database a first time. We obtain the following result

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

- All the itemsets of size 1 have a support of at least 3, so they are all frequent.

Recommendation Criteria

- Users need to login to view recommendations
- New users will initially have no recommendations
- If users have search & click history:
  - If user car searches does not have similar data with other users, the recommendation is based on his own search & click history sorted according to frequency
  - If user car searches have similar data with other users:
    - Retrieve all associated data and use *Apriori* to order based on the frequent *Car ID* given a minimum support.

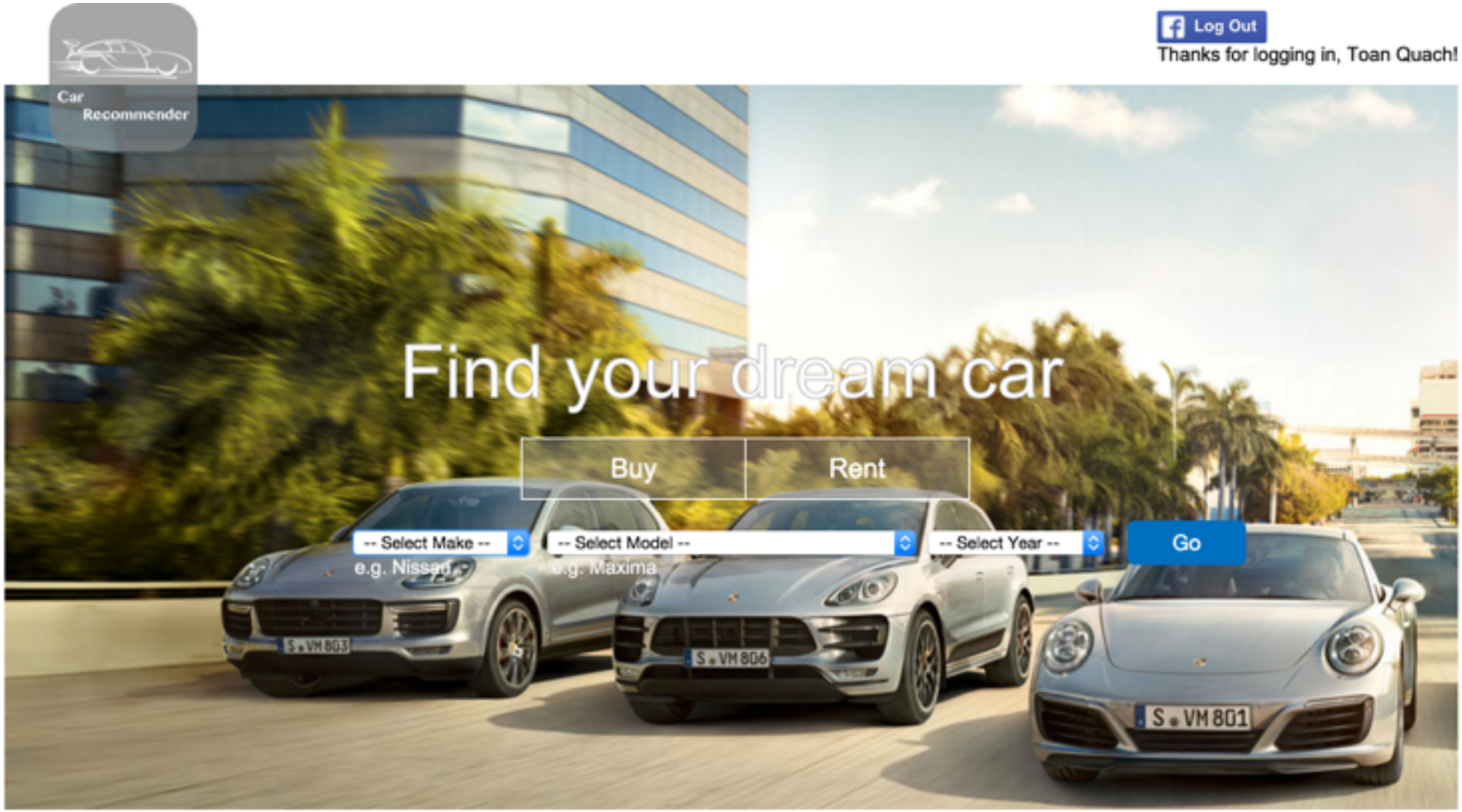
Tools & Technologies

- Java
- CSS
- JQuery
- Spring MVC
- Hadoop
- Hive
- Git / GitHub
- Facebook API
- Eclipse
- Edmunds Car Ratings API

Limitations

- Didn’t use the Spark & Pig in the project
- Need to enhance the recommendation system based on rating, reviews, ….
- The performance is little bit slow because of the connection to HDFS via Hive DDL, DML (e.g group by, insert statements…)
- Transaction handling is limited

Live Demo


Car
Recommender
Log Out
Thanks for logging in, Toan Quach!
Find your dream car
Buy
Rent
-- Select Make --
e.g. Nissan
-- Select Model --
e.g. Maxima
-- Select Year --
Go

Thank You!