Grocery Store Application with Item Recommendation

## 

## COMP4601

## For

## Professor Anthony White

## 

## By

## Md. Tanvir Islam 100969657

**Abstract**

This project seeks to design a grocery shopping website’s catalogue such that customers receive recommendations based on the items in the basket, maximizing sales and customer satisfaction while playing into the psychology of impulse buying.

Individual product data was extracted using a web crawler, which scraped all the products from a site called <https://fillmyfridge.ca/> and stored them MongoDB. This data was then piped into an android application depending on the search query made by the user using RESTful API calls.

**Introduction**

We are in the midst of a technological boom right now, catalysed by advancements in data collection; our exposure to incredible amounts of data increases every day, and with it so does our ability to measure, store, and predict the same-- giving us the capability to learn, experience and create without any limits. However, an exposure to such incredible amounts of data can be quite chaotic. With incredible amounts of data comes incredible amounts of choices and due to so many choices we can spend way too much time picking a certain data or worse pick something completely irrelevant to our needs. This is where recommender systems comes in handy. They provide clarity in our decision to make meaningful connections out of huge amounts of information. Good recommender systems will know exactly what the user has in mind and from trillions of data points give the user exactly what they are seeking out. An excellent example of a cutting- edge recommendation system would be Amazon’s product recommendations. It inspired me to develop my own version of this technology, for use towards a smaller business ventures, which pose their own set of challenges.

Using my application, the user can search for a product in the store from several product data extracted from fillmyfridge.ca. Upon receiving the products from the search result, the user can add the product to their cart or click on the cart for more details about the product. If the user decides to click on the card representing the product they will be shown further details about the product and other products recommended to them. These recommendations will be of two categories, substitutes and complements. Substitute goods are products that are similar to each other, thereby replacing one another on occasion, contingent on price and other factors. Meanwhile, complementary products are often bought in concert with one another. For instance, Coco Cola and Pepsi and substitutes, while chips and dip are complements.

**Background**

To build my system, I needed four main components, the client with which data is accessed, the server which serves the data, the data itself and most importantly the algorithm with which the recommendations could be made.

From thousands of products in the database the most relevant products need to be given back to the user according to their search, in order to do so I used a powerful indexing library call Lucene. Lucene is a full-text search library in java which makes it easy to add search functionality to an application or a website. It adds content to a full text index. It then allows you to perform queries on these indexes returning the most relevant documents depending on the query. Lucene is extremely fast because instead of searching texts directly it searches an index instead. It is similar to searching contents in a book through it’s indexes rather than going through every single page in a book looking for a word. This type of index is called inverted index, because it invers a page-centric data structure (page with words) to a keyword-centric data structure (words found in pages).

It is very important for a business to find the association rules between products that are bought together frequently. If the association rules can be found between products that are bought frequently together it can potentially increase the sales of a company, by better advertisements, discounts on one out of the two products or even create new products which combines the complementary products. The apriori algorithm is a very good association rule analysis algorithm which I have used for my complementary product recommendations.

The apriori algorithm measures the association rules using support and confidence quantifiers. The support of a set of items determines how popular that set is compared to all the available sets. It is the ratio of the frequency of the item set to all other item sets. The confidence says how likely item-b is purchased when item-a is purchased. To clarify the statement let’s say we have the following transaction sets {milk, cereal, break}, {milk, cereal}, {milk, butter}, the confidence of purchase of milk when cereal is bought is expressed as {milk -> bread} and the confidence is 2/3.

It is also very important to recommend user products that are similar to each other, for example, if a user wants to buy chocolate flavoured ice cream they may also want to try out other flavours like strawberry. In order to make such recommendations of similar products I use natural language processing on Lucene search queries in order to find the most relevant products similar to the current item for recommendation.

**Related Work**

For recommendation of products several other algorithms other than apriori could be used such one for example would be Item-to-Item collaborative filtering, an algorithm used by Amazon itself [1]. In Item-to-item collaborative filtering a table is created which consists of item pairings that customers tend to purchase together. This table is looked up when a client clicks on a new product they are purchasing to determine the other product that would go best with this product.

**Methodology**

In order to gather the data a web crawler written by me in Java was used to scrape through all the product catalogues from fillmyfridge.ca. Information from a product catalogue page was carefully extracted in order to facilitate the indexing and searching of user’s product requests.

**Discussion**

The goal of the project was to create an online grocery store application with a recommendation system which it does achieve, however there is a lot of room for improvement. The project’s biggest weakness its hunger for data. The biggest challenges I faced was to create a recommendation shop for a store that has nothing to recommend as there are no real and meaningful transactions in the database. Simulation of real transaction is difficult without an external source of data. The product catalogue extracted from the crawler will not always contain all the products which exists in the external transaction dataset. This results in creating substitute products which does not really reflect the real users buying patterns, this may provide faulty recommendations for complementary products.

However, the projects real strength lies in its search, interface and recommending similar items. The product catalog extracted from fillmyfridge.ca only had the title of the product, price and category it belonged to as texts that could be indexed using Lucene which made it difficult to retrieve relevant products due to the lack of enough text in the index. For example, a query with the text ‘green tea’ would return results for not only every document which included tea but also documents which had the word green in it, so finding all similar products related to tea would return irrelevant results. The strength of the project really lies in finding products that are alike by modifying search queries with natural language processing by identifying the nouns and adjectives and emphasizing the query with the noun part when looking up products in the Lucene index despite the lack of descriptive details about the product.

**Conclusion and Future Work**

I don’t know what to write in the conclusion….

In the future I would like to improve my recommendations given to users by adding an extra recommendation for a product just as the user is about to checkout with their items in their cart. This recommendation is going to use a user based collaborative filtering algorithm. The current items in the cart will be converted into a 1 x n vector where n is the total number of products in the database. From n products, products that exists in the cart will be represented by 1 and 0 otherwise. Transactions that have already been completed by other users will all have this vector representation of the transaction. The cosine similarity between the checkout vector and every other transaction vector is calculated. A cosine similarity value greater than 0.5 will indicate that the transactions are very similar to each other. A matrix will be created using the checkout vector and all other similar vectors and user based collaborative filtering will be done on this matrix to find more items that the user can purchase with his current selection of items.

In the future I aim to introduce rating system to the items and create a user base where user’s can rate that they have purchased. Adding these two functional features to the application will provide a greater opportunity to improve product recommendations as algorithms such as item based collaborative filtering can then be implemented.

**References:**

* http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf