INFORMATION RETRIEVAL TERM PROJECT REPORT

Group Members

- Haritabh Singh (12CS10023)
- Manav Kedia (12CS10057)
- Prithwish Mukherjee (12CS10058)
- Soham Dan (12CS10059)
- Agnivo Saha (12CS10062)
- Anurag Anand (12CS30006)
- Biswajit Paria (12CS30009)

INTRODUCTION

In this project we have developed a system for Nikon cameras through which we can return search results in response to a user's free form queries. Special use cases like refined searches based on categories of products have also been implemented. Ranking methodology is novel and tailored to the particular need of such a search system for products. In addition a recommendation system has also been built to return a list of recommended products for the user. Apache Nutch has been used for crawling and AngularJS and NodeJS has been used for the frontend and backend respectively.

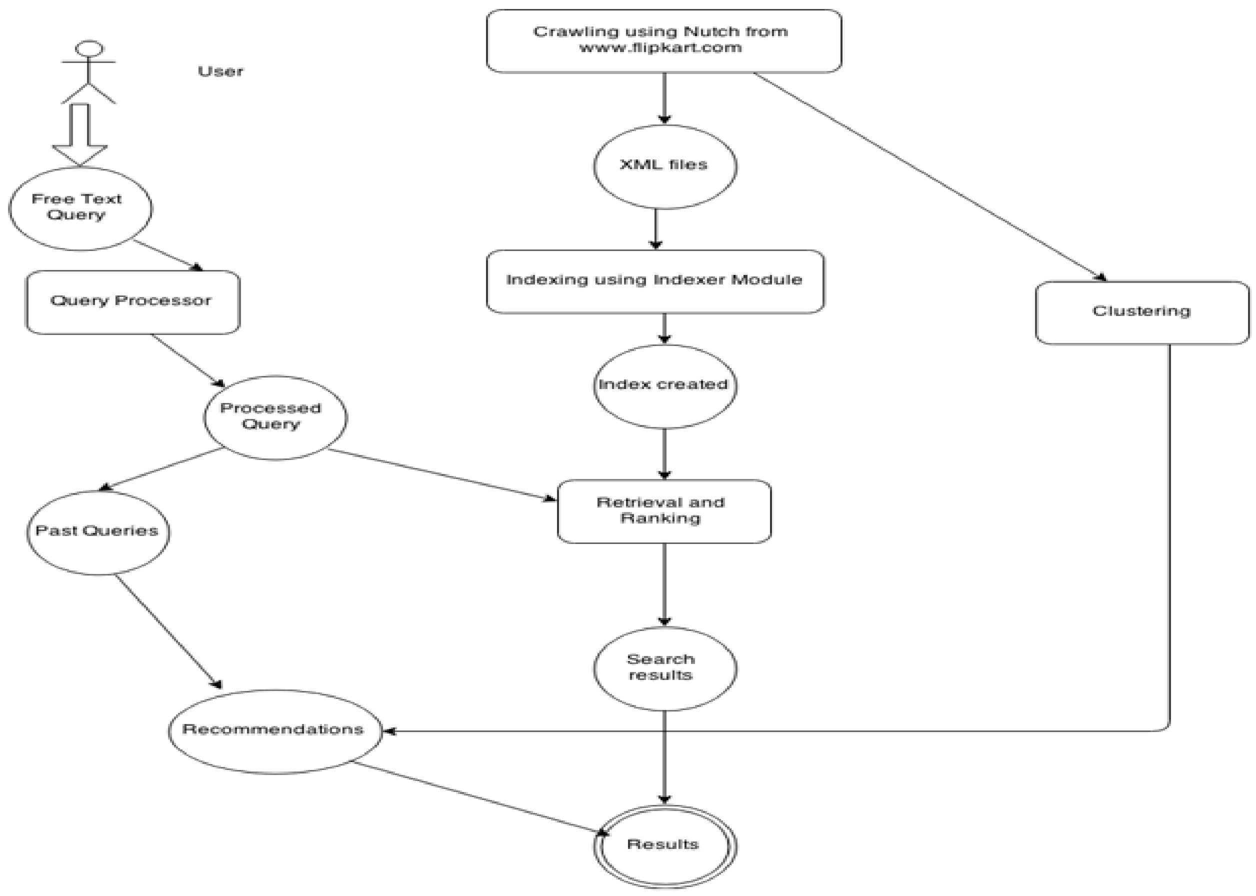


Fig 2. Complete overall Workflow

USE CASES

Broadly 2 types of use cases have been dealt with.These are explained below:

Search : We have supported free form queries. Free form queries basically means that the queries are not structured or rule based. They are expressed in the natural language as we speak or write. From the search queries, attributes have been extracted for better ranking and recall. This is because the attributes if matched will return specific results and thus we shall get a high recall. Additionally this is meant to match the user needs and hence precision too will be high.

Recommendation : This module is based on the past search history of users. For this a matrix of search history is maintained which we have elaborated in the later part of the report. Based on this matrix and certain other parameters the system returns to the user a list of recommended products which the user might be interested in.

CRAWLING

The official sites of Flipkart and Nikon have been used for crawling the data needed for our system. http://www.flipkart.com and www.nikon.co.in/ were crawled. Apache Nutch is an open source Web crawler written in Java. By using it, we can find Web page hyperlinks in an automated manner and also reduce lots of maintenance work, for example checking broken links, and create a copy of all the visited pages for searching over. We used Nutch for crawling the sites.Firstly we have to set a seed list of URLs to crawl. A URL seed list includes a list of websites, one-per-line, which nutch will look to crawl.
Nutch data is composed of:

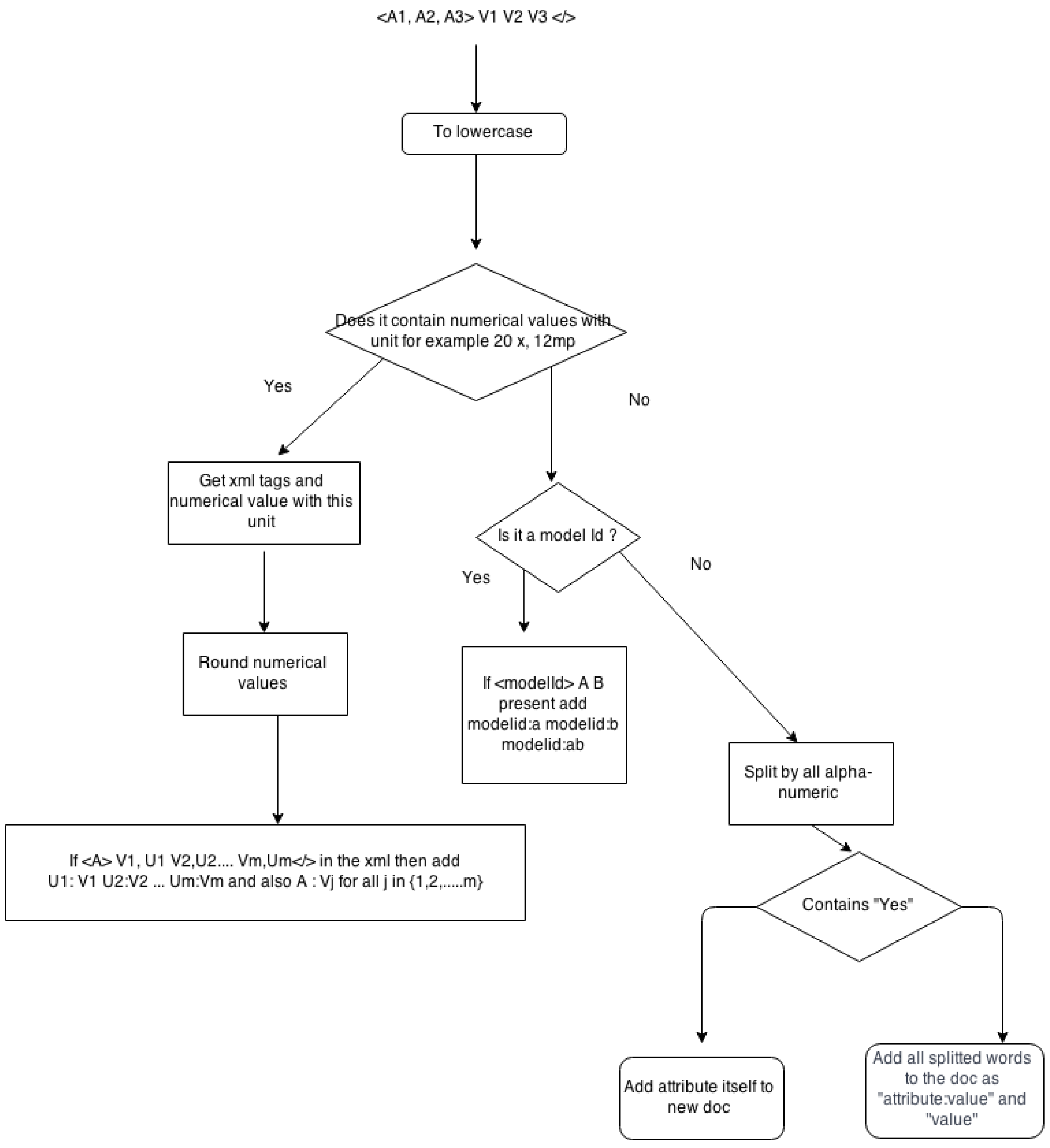
1. The crawl database, or crawldb. This contains information about every URL known to Nutch, including whether it was fetched, and, if so, when.
2. The link database, or linkdb. This contains the list of known links to each URL, including both the source URL and anchor text of the link.
3. A set of segments. Each segment is a set of URLs that are fetched as a unit. Segments are directories with the following subdirectories:
    - a *crawl_generate* names a set of URLs to be fetched
    - a *crawl_fetch* contains the status of fetching each URL
    - a *content* contains the raw content retrieved from each URL
    - a *parse_text* contains the parsed text of each URL
    - a *parse_data* contains outlinks and metadata parsed from each URL
    - a *crawl_parse* contains the outlink URLs, used to update the crawldb

Statistics for crawling :

Time taken to crawl the data : 6 hours
Number of the crawled documents :7312
Size of raw data : 600 MB
Size after indexing of raw data :17.34 MB

INDEXING

For indexing firstly the attribute value combination was converted to lowercase entirely. Then the steps as specified in the flow chart were carried out.If numerical values are present with units we get the xml tags for them. Otherwise we check if it is a model id and accordingly split by alphanumeric characters and then index. For further clarification please refer to the flowchart below.
Example :
color:Blue ratingValue:4.042 price:Rs17,950 Brand:Nikon ModelID:CoolpixAW100



<A1, A2, A3> V1 V2 V3 </>
To lowercase
Does it contain numerical values with unit for example 20 x, 12mp
Yes
No
Get xml tags and numerical value with this unit
Is it a model Id ?
Yes
No
Round numerical values
If <modelId> A B present add modelid:a modelid:b modelid:ab
Split by all alpha-numeric
If <A> V1, U1 V2,U2.... Vm,Um</> in the xml then add U1: V1 U2:V2 ... Um:Vm and also A : Vj for all j in {1,2,.....m}
Contains "Yes"
Add attribute itself to new doc
Add all splitted words to the doc as "attribute:value" and "value"

Fig 3. Flowchart on Indexing

SEARCH

As we have already mentioned free form text queries are supported for searching. Matching just the numerical values for the search results is insufficient. In that case it would return any field having the same numerical value (please see the example below) . This is why we compound the numerical values along with their units to produce the tokens. Also the numerical values are rounded properly to increase recall. Note that in the indexing an identical rounding of values has to be carried out to ensure match occurs. Our retrieval methodology tries to ensure that attribute value pairs play a role.

For example Consider the case where the user supplies in his query Rs 5000 , this would try to find cameras with cost = Rs 5000 rather than cameras with resolution 5000 x 7000

This is why we need to tokenize the numerical value along with its unit.
Example :   Mirrorless Black Colour 20 mp

QUERY PROCESSING

For query processing we use a modified form of the query given by the user called the transformed query. Numerical tokens are separated and bigrams are formed between their values and word tokens and also their values and their units. Bigrams of work tokens are also constructed. All these bigrams are then collected together to form the transformed query.Please see the flowchart below for further clarification.

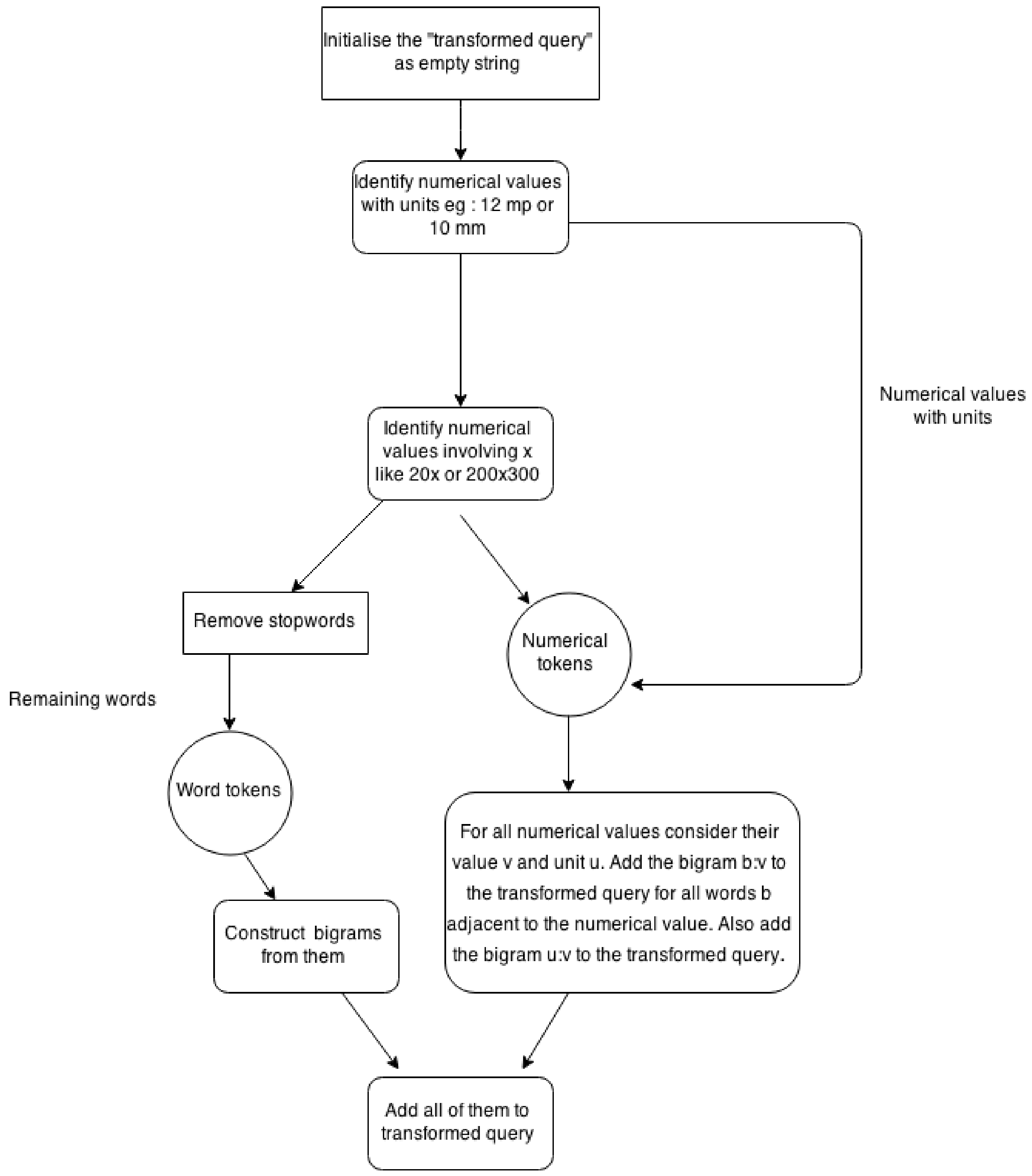

Initialise the "transformed query" as empty string
Identify numerical values with units eg : 12 mp or 10 mm
Numerical values with units
Identify numerical values involving x like 20x or 200x300
Remove stopwords
Numerical tokens
Remaining words
Word tokens
For all numerical values consider their value v and unit u. Add the bigram b:v to the transformed query for all words b adjacent to the numerical value. Also add the bigram u:v to the transformed query.
Construct bigrams from them
Add all of them to transformed query

Fig 4. Flowchart on Query Processing.

Example from above search query:
black:mirrorless color color:20 mp:20 color:black mirrorless black:color 20:mp 20:color black mirrorless:black

RANKING

If a token in the search query matches a token in the index we must rank based on the significance of this match.An ordinary tf-idf measure is quite obviously invalid in our case. We therefore worked out a ranking methodology which is suitable for our case. Ranking is done based on match with attributes which are previously stratified into classes. Attributes were categorised into 3 types depending on their characteristics .These characteristics included whether they had a value associated or not, whether they were a particular defining characteristic or not .Separate scores were assigned to each of these types.

Depending on the match with query terms each document was assigned a score based on which type the matching token was present in.

Score computation : $\Sigma(score_{term}\alpha + (1-\alpha)\ rating_{doc}) * (f(term))$

where $f(i) = 1\ if\ if\ term\ i\ is\ present\ in\ query$

$else\ f(i) = 0$

Documents were ranked based on these scores

We took α= 0.8 after experimenting with the search results.

Example :   Category with score 3 : Model number, gps, Wi fi, etc.

Category with score 2 : color, megapixel,resolution ,etc.

Category with score 1 : other additional features.

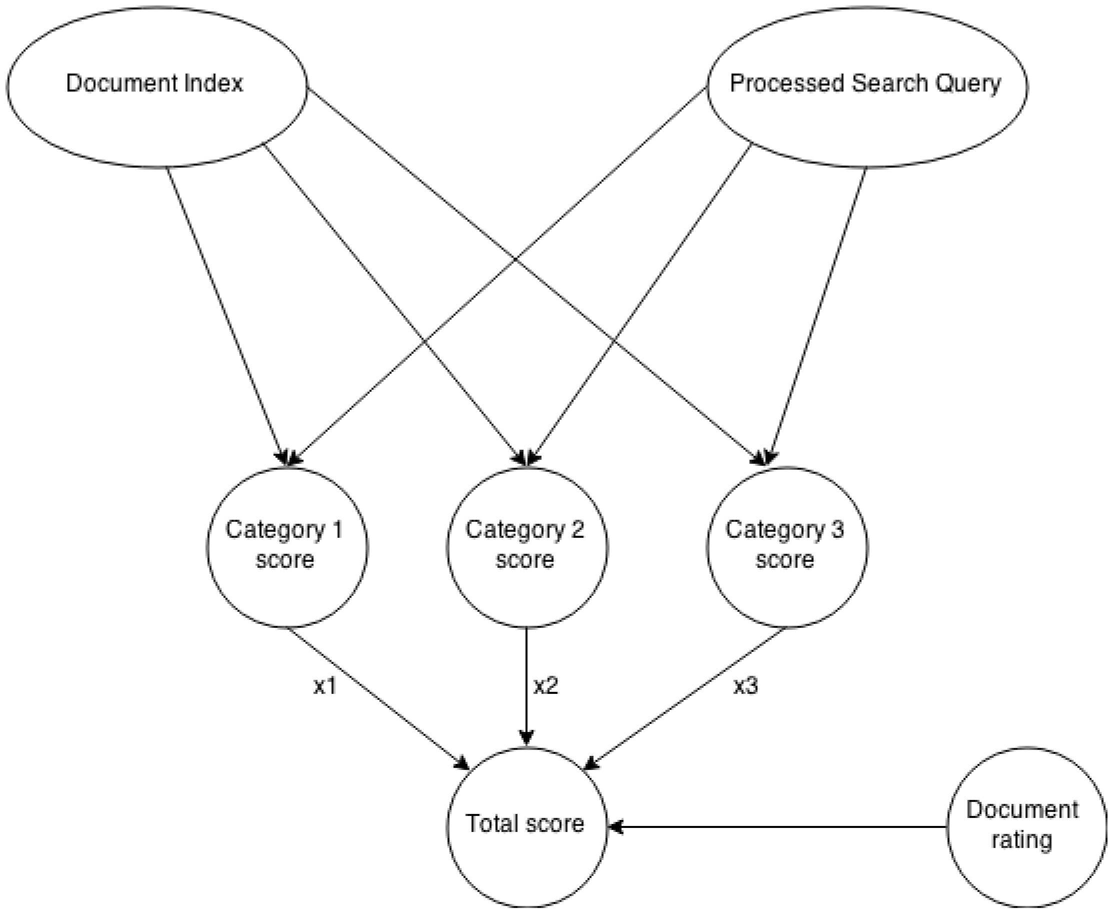


Fig 5. Flowchart on Ranking System

RECOMMENDATION

For the recommendation system there are some associated data structures.
These are a mxn matrix S and a vector D for each document (product in this case).
Also there are two parameters a and b which are the degradation factor and the search rank respectively.
Let S_{mxn} denote the search results matrix which is the past m search results with top n results for each. Additionally we have a vector $D_N = < 0, 0, \dots , 0 >$ denoting the weight vector for each document.For each element S_{ij} in S we do $D_{S_{ij}} \mathrel{+}= a^i b^j$, where $a,b < 1$, denote the degradation factor for search age and search rank respectively.
We experimented to see that $a = 0.6$ and $b = 0.8$ gave good recommendations.
The logic behind this is that as the search age increases the search becomes less relevant for recommendation. This is indeed true as we want to recommend to the user current trending products.
For recommendation the top n document Ids with highest weights are selected and displayed to the user .

REVIEW

We had extracted all the past reviews given for a particular product from the websites we had crawled.These mined reviews from the crawled data are neatly organized and displayed to the user along with their rating values.
These are used to find unknown ratings of products using regression on the sentiment analysis values as input.

SENTIMENT ANALYSIS

From the reviews it is possible to assign an opinion of the user regarding the product . This is the basis for sentiment analysis of the user reviews. We used the NLTK Library for the sentiment analysis. The sentiment classifiers are trained on Wordnet corpus. Positive and Negative sentiment values were found for all available reviews using the Python NLTK Library.

Using the data of available ratings and least squares regression using a second degree polynomial the missing rating values were predicted.

RMSE = 0.4188

Example : This camera is not so bad as described by others. Its GPS function is excellent. Picture quality also good and colour are realistic not like over burned colour. Its only problem is taking picture of any moving objects without flash are became blurred….

Polarity:

0.3 positive

0.7 negative

SYSTEM ARCHITECTURE

The architecture of the system is a Model View Controller type. Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

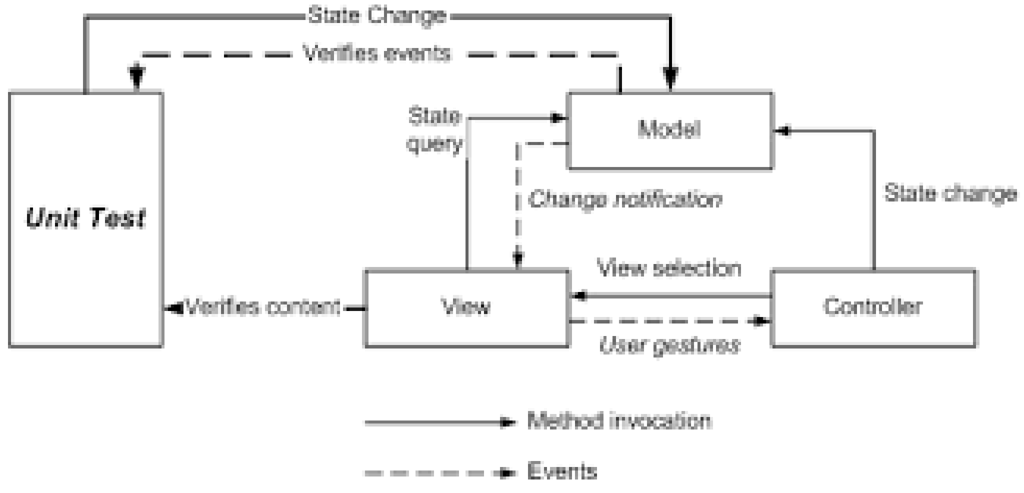
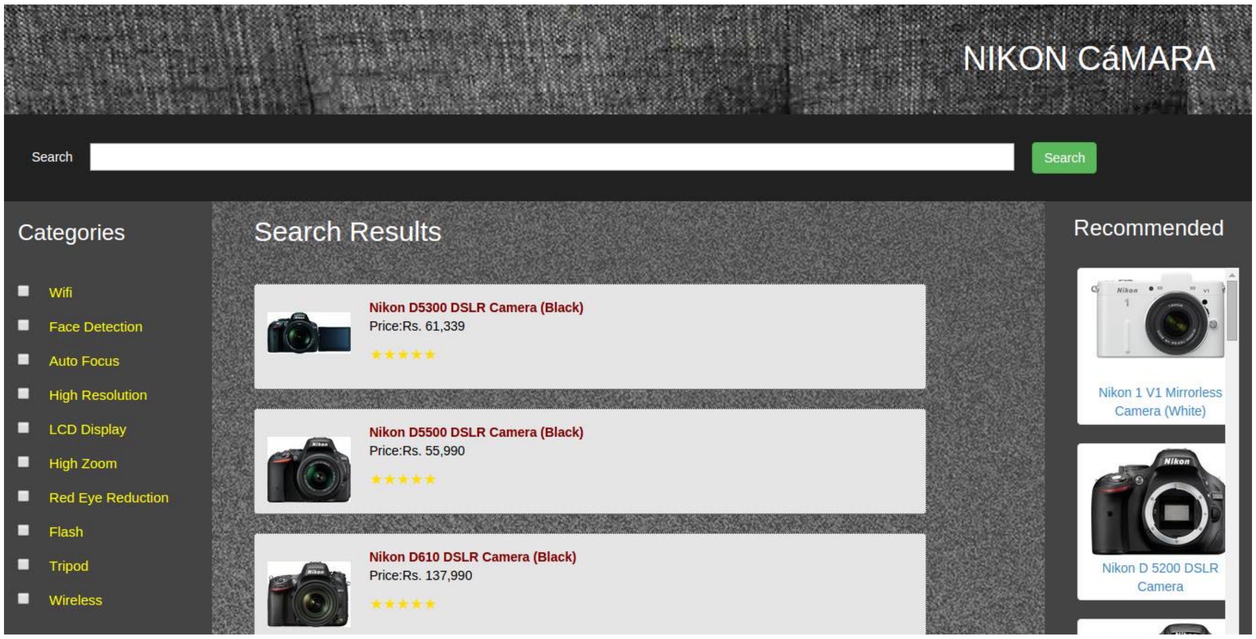


Fig 1. MVC Architecture

GUI FRONTEND



Screenshot of the working system.

For the frontend HTML , CSS , Angular js, jQuery was used.

The graphical user interface is simple and extremely user friendly.It is easy to use and navigate. It is lightweight. There is a panel on the left to refine the search based on several specific categories. The search results are displayed in the center panel.
In the right panel recommended products based on past searched of users are displayed.

GUI BACKEND

Node js has been used for the backend. Node js backend serves the pages for the front end and handles GET and POST requests for the search and recommendation results.
With Node.js a group of small applications is developed instead of one large application, this enables a change to be made or new functionality to be added without requiring changes to be made deep inside the entire code-base.
It invokes the Java and Python code with search query and returns the results back to front end for display to the user.