Information Retrieval Term Project

Camera search system using Nutch as a crawler

Group Details

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

Use Cases

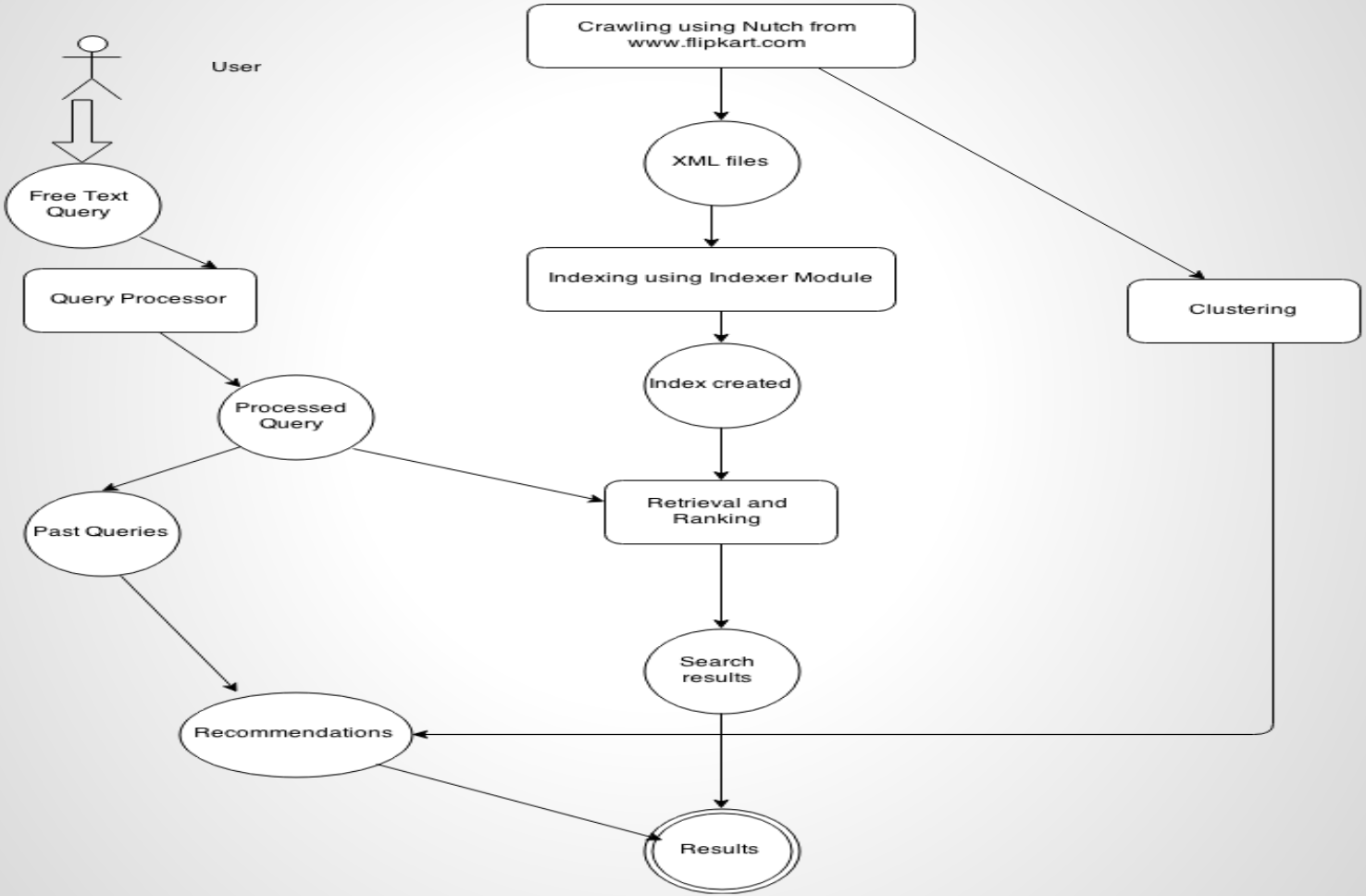
SEARCH

- Free form text queries.
- Extraction of attribute features for better ranking and recall.

RECOMMENDATION

- Based on the users past search history the system recommends cameras which might suit the users needs

Complete Workflow



Crawling

- **Websites :**

  www.flipkart.com

  www.nikon.co.in

- We used Nutch for Crawling
- Initial seed urls were chosen and the collection of url were expanded
- 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

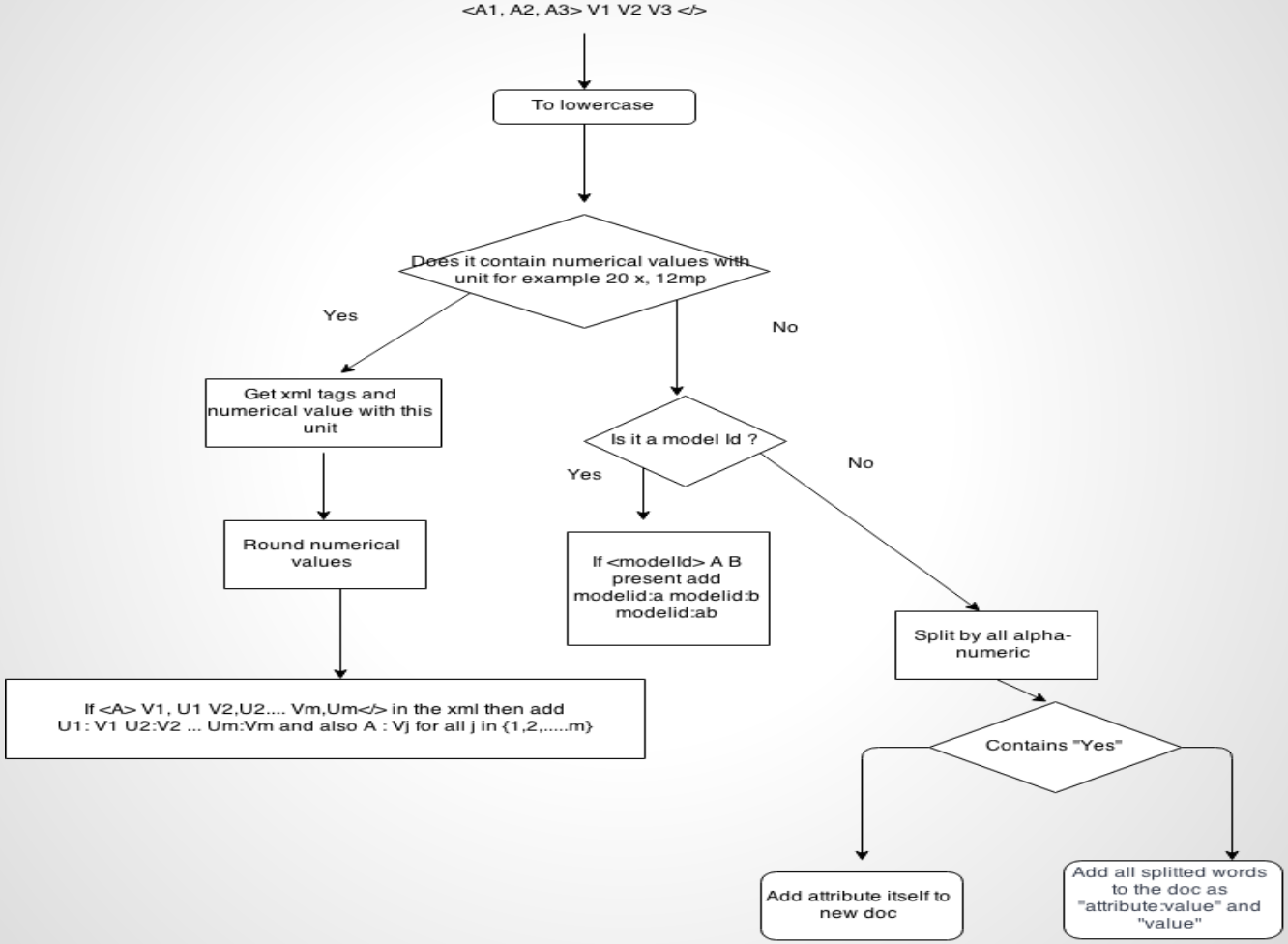
User Reviews

- Review were mined from the past data
- These reviews are neatly organized and displayed to the user
- These are used to find unknown ratings of products using regression

Sentiment Analysis

- Positive and Negative sentiment values were found for all available reviews using the Python NLTK Library.
- A substantial number of rating values missing
- Using the data of available ratings and least squares regression using a second degree polynomial the missing rating values were predicted. RMSE = 0.4188

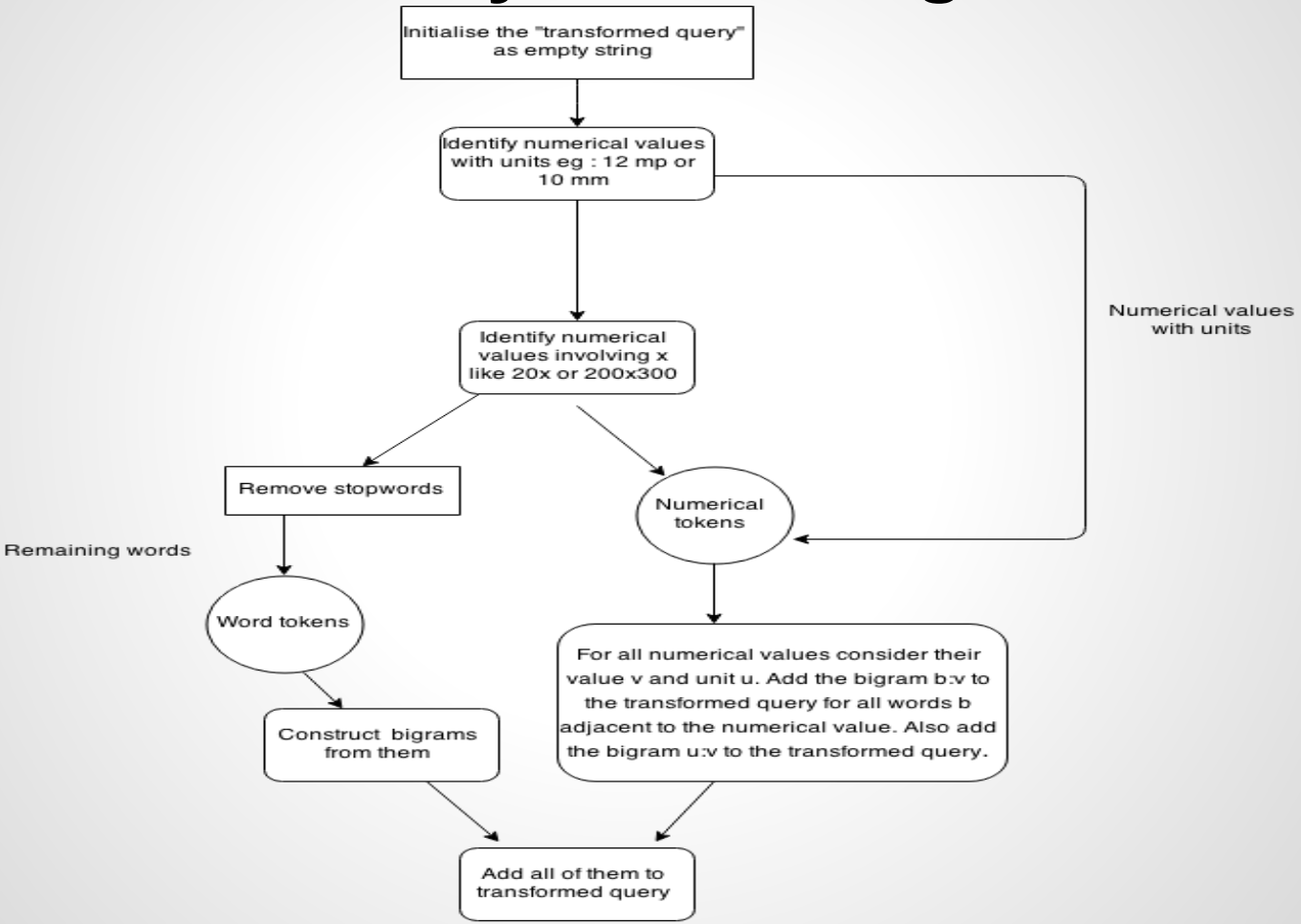
Pre-processing of the Crawled Data



Search

- Free text queries are supported.
- Numerical values along with their units have been tokenized and rounded properly to increase recall.
- Retrieval method tries to ensure that attribute value pairs play a role.
- For e.g Rs 5000 would try to find cameras with cost = Rs 5000 rather than cameras with resolution 5000 x 7000

Query Processing



Examples

- Indexed terms:
  - 1st type:
    manual focus manual exposure wi fi connectivity wireless connectivity tripod flash red eye reduction self timer
  - 2nd type:
    id:aw1 lcddisplay:lcd lightsource:active fps:30 otherresolution:640x240 fps:400 videodisplayresolution:30 megapixels:14
  - 3rd type:
    illuminator waterproof tft lcd daylight arabic bengali chinese panorama lithium ion battery
- Initial Query : Mirrorless Black Colour 20 mp
- Processed query: black: mirrorless color color:20 mp:20 color:black mirrorless

  black:color 20:mp 20:color black mirrorless:black

Ranking

- Attributes are categorised into 3 types depending on their characteristics.
- Separate scores were assigned to each of these types.
- Depending on the match with query terms each document was assigned a score viz. 1,2,3
- Documents were ranked based on these scores

Score Computation Formula

Score computation :

$$\Sigma \left(score_{term} * \alpha + (1-\alpha)\ rating_{doc}\right) * (f(term))$$

where $\begin{cases} f(i)=1 \text{ if if term i is present in query} \\ \text{else } f(i)=0 \end{cases}$

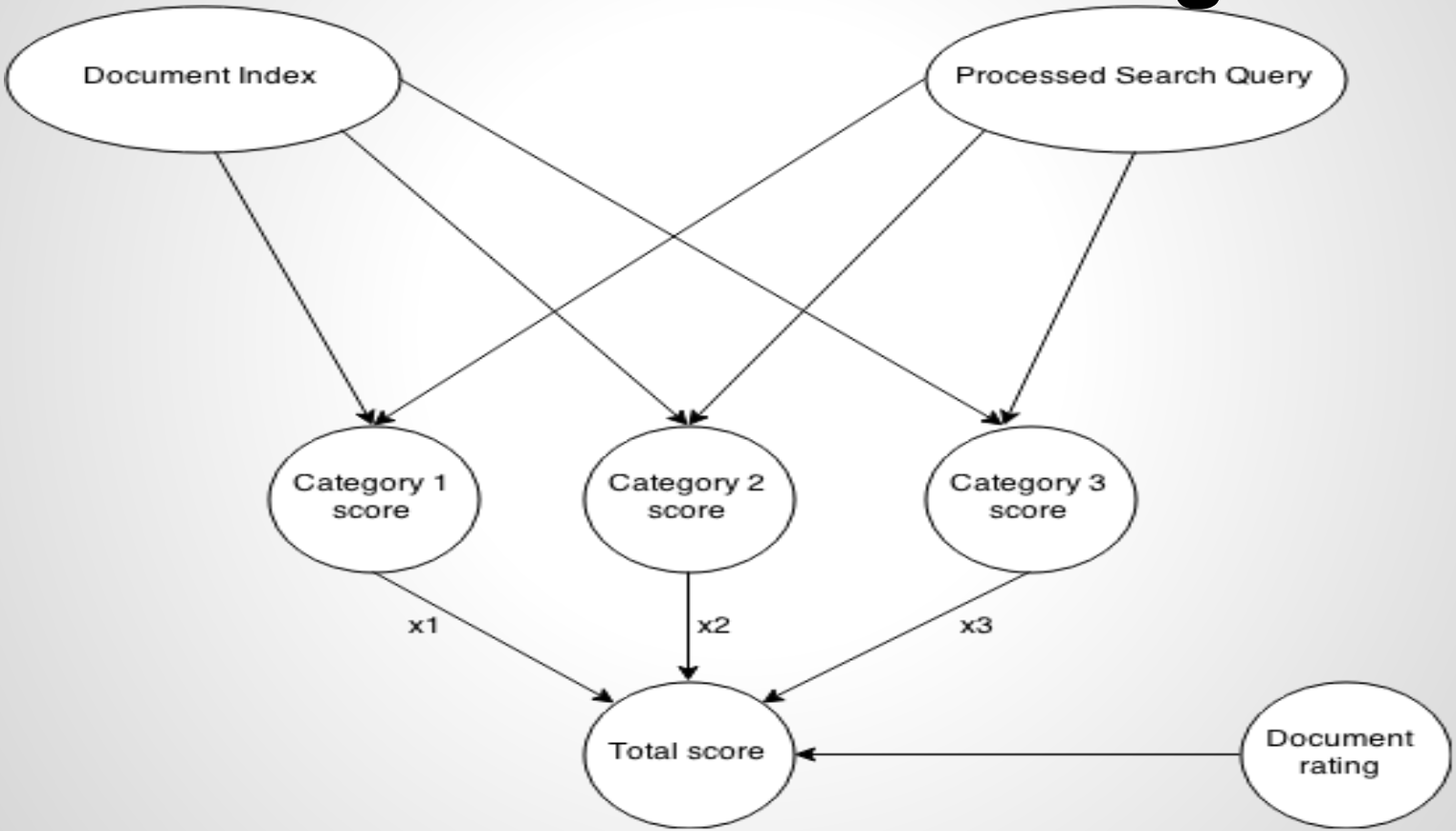
Category with score 3: Model number, gps, wi fi ,etc.

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

Category with score 1: other additional features.

We took experimented and took $\alpha = 0.8$

Flowchart for Ranking



Recommendation

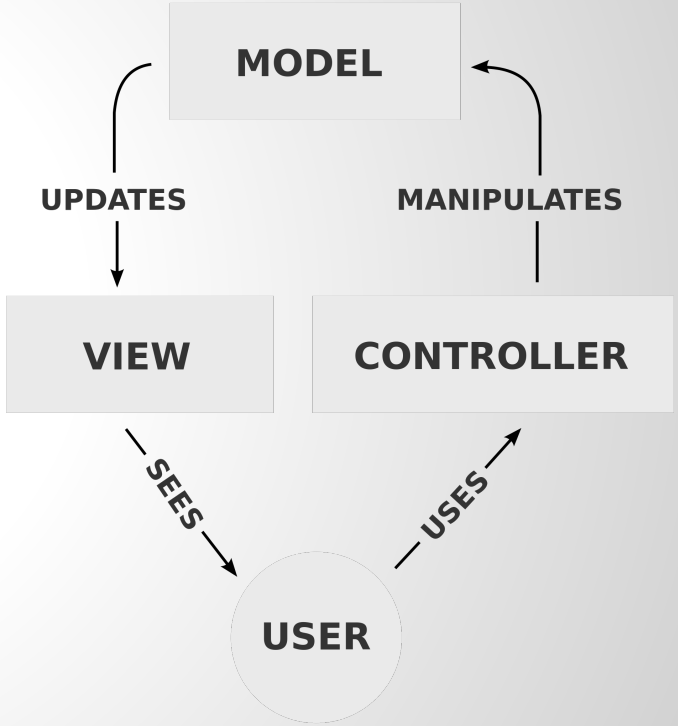
- Let S_{mxn} denote the search results matrix which is the past m search results with top n results for each. Additionally we have a vector $D_N = < 0, 0, \ldots , 0 >$ denoting the weight vector for each document.
- For each element S_{ij} in S we do $D_{Sij} \mathrel{+}= a^i b^j$, where $a,b < 1$, denote the degradation factor for search age and search rank respectively.
- We did experiments and found good results for a = 0.6 and b = 0.8.
- We also cluster the available documents into 10 clusters using their attribute values and text descriptions.
- For recommendation we sort sort the cameras according to their weights. Then for each camera we return the highest rated camera belonging to its cluster which hasn't been selected in the current recommendation.

Clustering

- Attributes of each xml were extracted and numerical values were parsed.
- Numerical values along with string values were used to cluster documents to 10 clusters using weka EM algorithm implementations.

System Architecture

- The architecture of the system is an MVC (Model View Controller) Architecture
- There is a data model which is represented in the view via the controller.
- Any change in the view will be reflected in the model



Graphical User Interface

FRONTEND

- HTML
- CSS
- AngularJS
- JQuery

BACKEND

- NodeJS
- Java

Front-end

- Simple user friendly GUI
- Easy to use and navigate
- Lightweight
- Panel on left : Refined search based on categories
- Sorted search results based on ranking

in centre.

- Recommendation of products user might find interesting

based on past queries

Back-end

- Node js backend is serves the pages for the front end and handles GET and POST requests for the search and recommendation results.

- It invokes the Java and Python code with search query and returns the results back to front end for display.