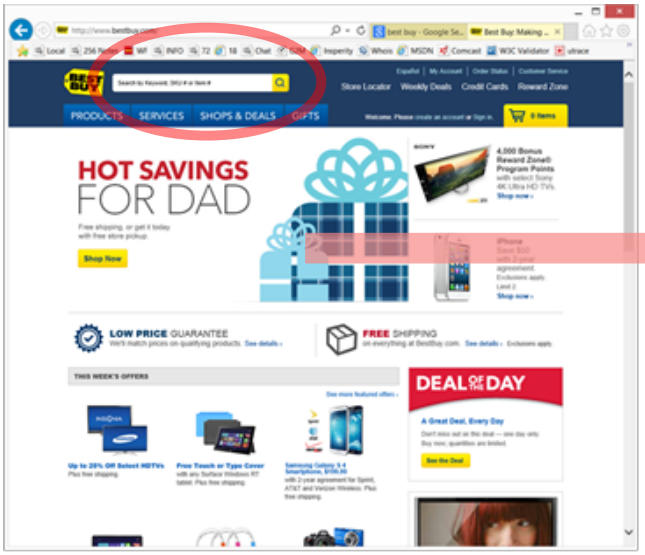
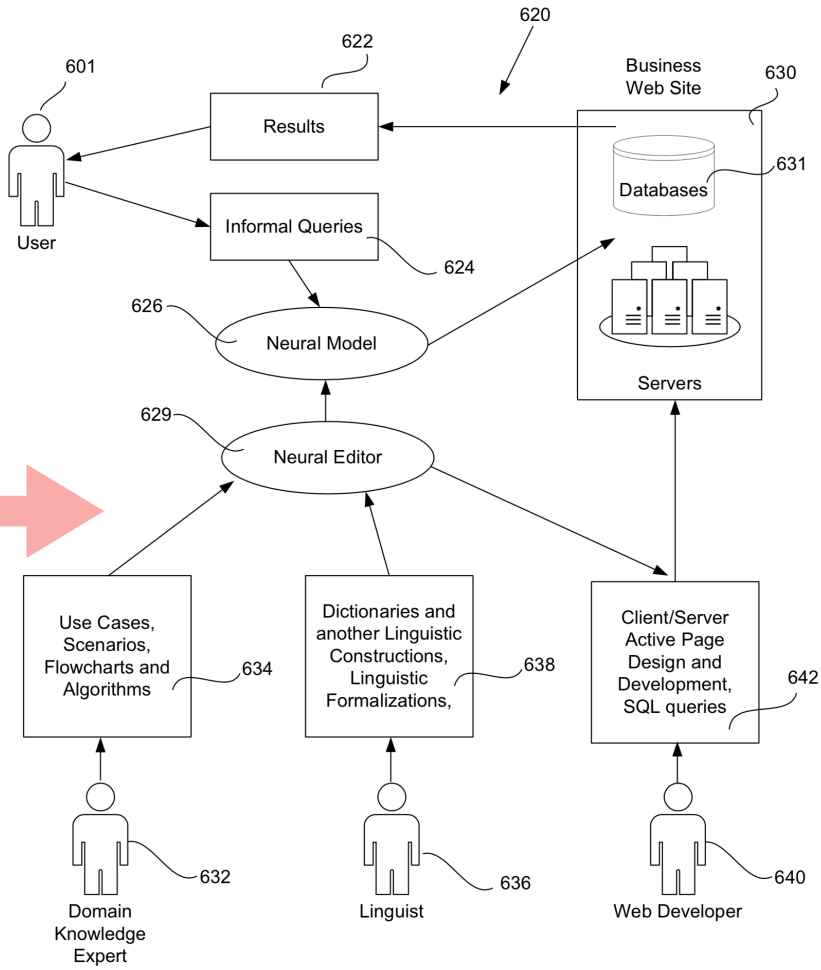
ChatBot in 7 Days

How to use Neural Networks to create Chatbot in a week

Development Diary: February 22, 2018 - February 28, 2018

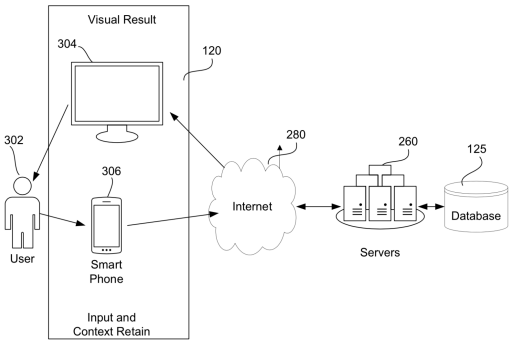
Best Buy

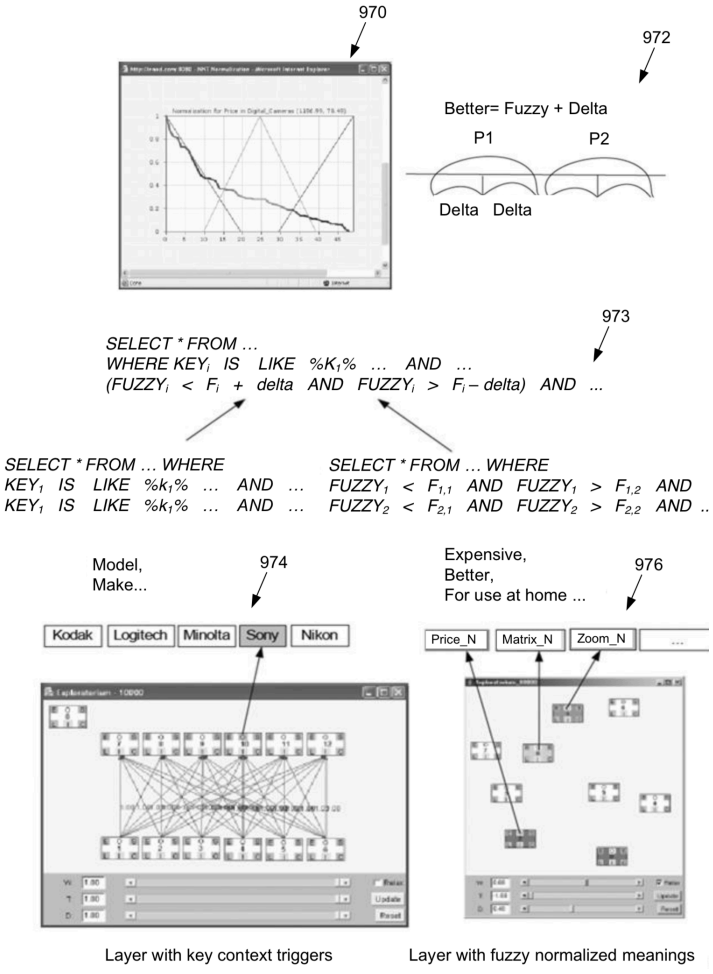




Introduction

In this project we will create a ChatBot or Virtual Assistant, capable to run in multi channel environment (chat, voice, PC, smartphone, TV) by using neural networks NNOD SDK and very basic development resources. The goal is to show how easy is the process of building interactive contextual search Assistant when using Neural Networks based Expert System. As an example we will use BestBuy website and its Development API. The process will takes one man-week, and I will keep the diary with daily report.





Day 1

Project preparation. Agile vs Waterfall. Data Source.

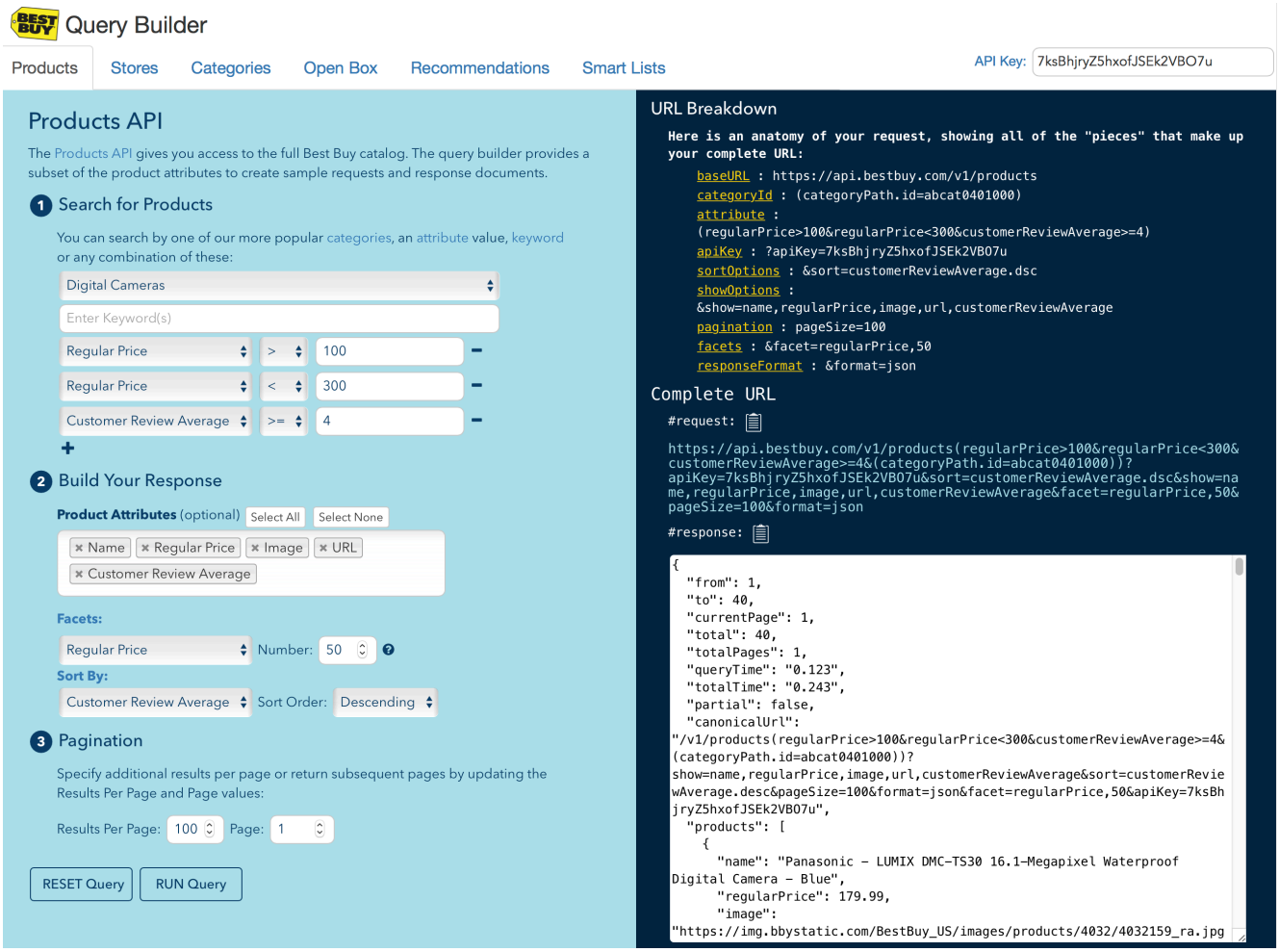
Before to start any projects, somehow, the decision about planning and final goal must be made. But I will be the only one working on it during the next 7 days, so all traditional function: management, architectural design, UI, front and back programming, QA and few others, will be implemented inside the one man's mind. Personally, I prefer the Waterfall model, but considering that I will have minimum possible resources allocated to this project, I've decided to use Agile as my personal method to communicate. Which means that "virtual manager", "virtual architect", etc. will "virtually" interact in my mind representing the model of real team cooperation.

Learning Best Buy API.

The very first step - registration on BestBuy Development site and obtaining API key. To do this we registered on https://developer.bestbuy.com and feel application form to obtain API Key. This Key required to send requests to BestBuys database.

The next step is to lear and experiment with BestBuy API (BB API). It takes some time to understand the structure, component, formats of API. The most important for us is Query Builder - http://bestbuyapis.github.io/bby-query-builder/#/productSearch Most of the time we spent to understand the structure of JSDON queries and responds.

For example, informal request - "Show me cameras around $200 with the best reviews" can be reformulated in formal request:



At the same time, on the background, we start architectural design and preparation for presentation of responses in more dynamic visual form.

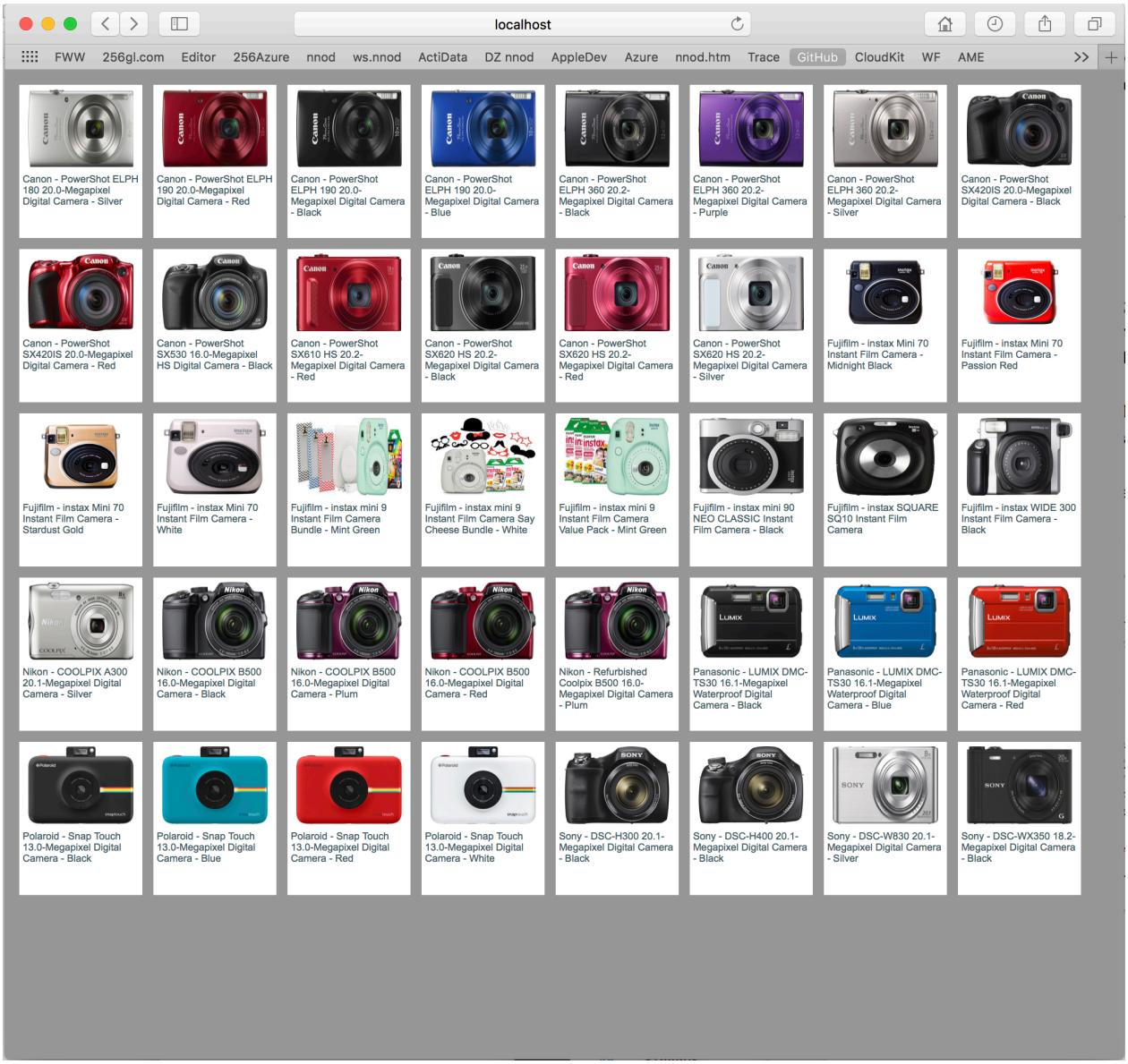
To make this solution generic as possible, we will use basic Web Server and will build interactive chat page using only standard JavaScript.

Our Neural Network API published as Open Source under GNU license in GitHub, and we will add this solution into the same GitHub depository.

Local Development Environment.

The whole process will be made in Mac OS but there are no any differences in the implementation for Windows. We will use Apache Web Server on Mac and IIS Web Server on Windows.

To make the very first call to BB API we can start with simple PHP request (potentially one of the next versions may require server integration):



```
<body style="background-color:#999999;">
<?php
      $url = "https://api.bestbuy.com/v1/
      products(customerReviewAverage%3E=4&regularPrice%3E150&regularPrice%3C250&(ca
      tegoryPath.id=abcat0401000))?
      apiKey=xxxxxx&sort=name.asc&show=manufacturer,name,image,url,regularPrice&fac
      et=regularPrice,50&pageSize=100&format=json";
      $data = file_get_contents($url);
      $json = json_decode($data, true);
      $array = $json["products"];
      $db_counter = 1;
      foreach ($array as $obj) {
          $db_counter = $db_counter + 1;
          echo "<div class=\"thumbnail_wrapper\"><table cellspacing=\"0\"
      cellpadding=\"0\"><tr>\n" .
          "<td id=\"td_thumbnail_" . $db_counter . "\"
      class=\"td_thumbnail_image\">\n" .
          "<a href=\"#\"><img src=\"" . $obj['image']  . "\" height='70' width='95'
      " .
          "\" onclick=\"javascript:openThumbnailLink('" . $obj['url'] . "'); return
      false;\"></a></td>\n" .
          "</tr><tr><td id=\"td_thumbtitle_" . db_counter . "\"
      class=\"td_thumbnail_title\">\n" . " " . $obj['name'] . "\n" .
          "</td></tr></table></div> \n\n";
      }
      ?>
</body>
```

The result:

Day 1 conclusion.

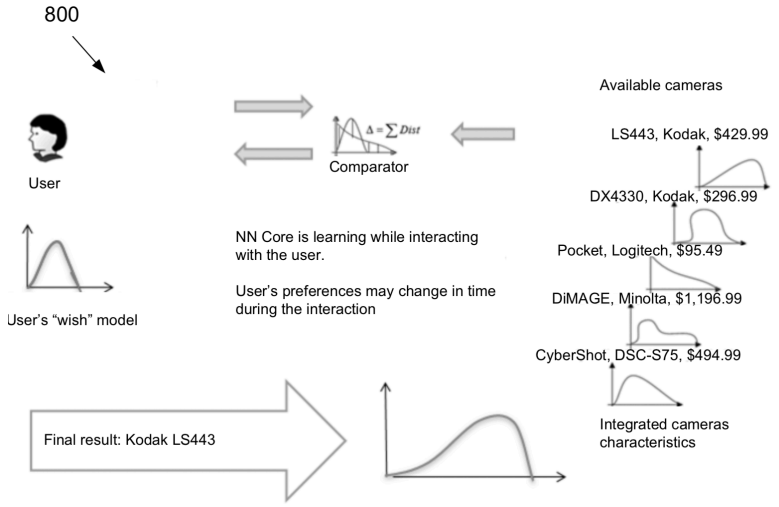
BB API Key obtained and we have full access to BB DataBase. The very first Web Page with implementation of requests to BB API built. Clear understanding of BB API now allow to start building Linguistic model for interactive search.

Day 2

Architectural/Marketing.

What is the value of my chat bot? - one of the most important questions must be clearly answered before and during this development. What is the criteria we should put to measure the project advantages from the similar? In our case I will use cost of service and user's information demand satisfaction.

When a user gets an idea about buying something, he/she needs to gather some information about the product. And before the very last moment when sale completed, the information for decision making will be collected from many sources. Web, personal notes, social networks and finally human experts are the network of participants in interactive process of choosing the best product from many available on the market.



Basically the process includes collecting data from different sources, compare result and repeat the process until the threshold level for the decision reached.

The cost of this process has two components: User's cost, which includes the time and potential difference if the item is wrong, or the same item can be purchased cheaper. And sale's coat, which includes the cost of consulting, and potential cost of loosing the client if he/she decided to switch to different store because of whatever reason.

We will use Best Buy Web site as one of the sources for measuring success. Criteria #1 will be based on the time required for the user to find enough information about the product and make decision to move forward and visit Best Buy or to switch to another retailer.

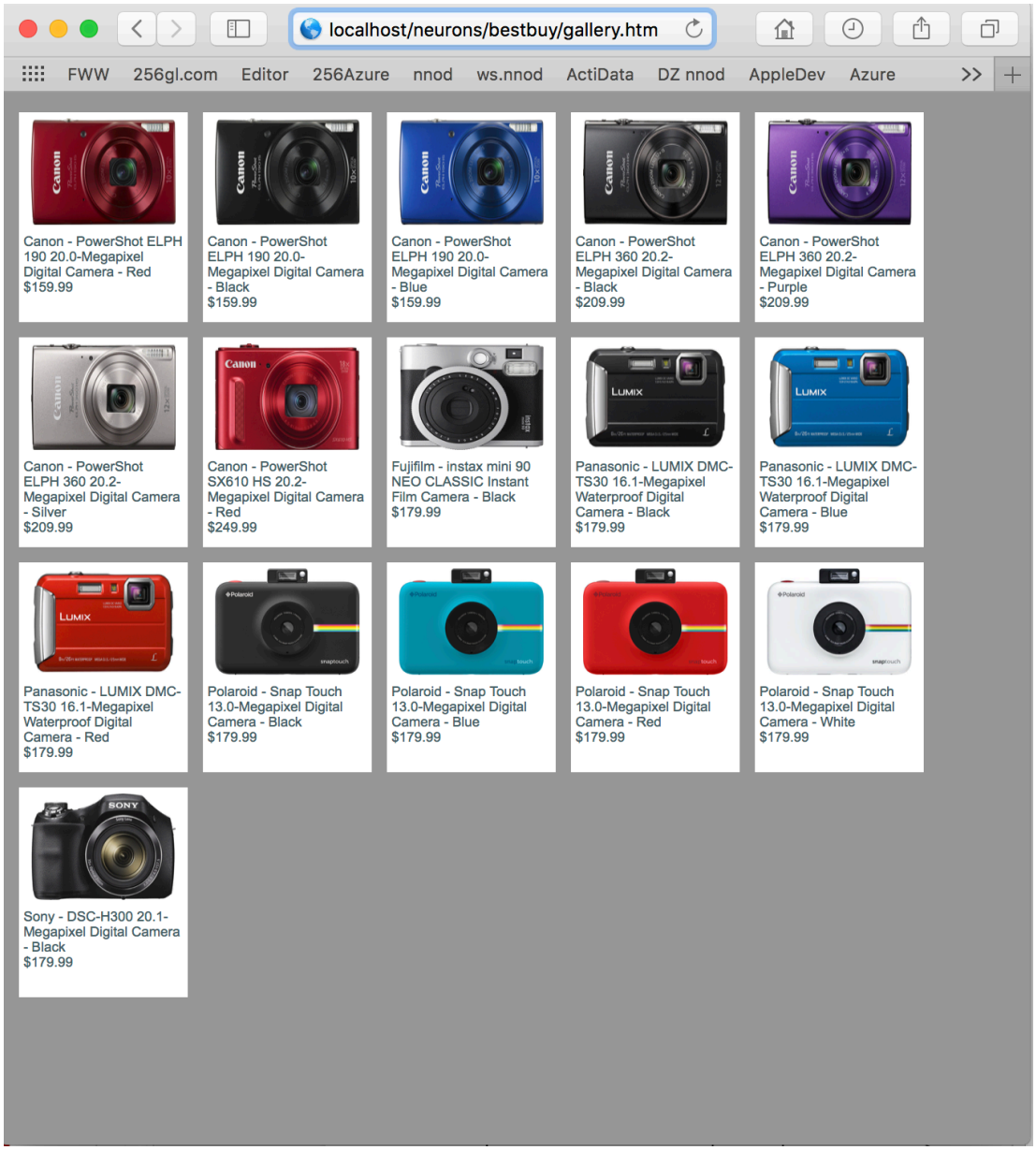
PHP -> JavaScript.

Even PHP is very popular and widely supported language, wherever it is possible we will try to use JavaScript for the reason of minimization of support and scalability. If our solution will be fully JavaScript based and will not require any external server support, but only BB API, that's mean we automatically solve the problem of scalability.

Changing PHP to JavaScript is relatively easy:

```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="viewport" content="minimum-scale=1.0, maximum-scale=5.0, user-
scalable=yes, initial-scale=1.0" />
    <meta name="viewport" content="user-scalable=yes">

    <title>256gl NNOD</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
    <script type="text/javascript">
        function openThumbnailLink(url) {
            window.open(url, "_blank", "toolbar=no, scrollbars=yes, resizable=yes,
top=10, left=10, width=800, height=800");
            return true;
        };
    </script>
</head>
<body style="background-color:#999999;">
    <div id="demo"></div>
    <script>
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                var myObj = JSON.parse(this.responseText);
                var result = myObj.products;
                var x = "";
                for (i in result) {
                    x = x + "<div class=\"thumbnail_wrapper\"><table
cellspacing=\"0\" cellpadding=\"0\"><tr>\n" +
                        "<td id=\"td_thumbnail_" + i + "\"
class=\"td_thumbnail_image\">\n" +
                        "<a href=\"#\"><img src=\"" + result[i].image + "\"
height='70' width='95' " +
                        "\" onclick=\"javascript:openThumbnailLink('" +
result[i].url + "'); return false;\"></a></td>\n" +
                        "</tr><tr><td id=\"td_thumbtitle_" + i + "\"
class=\"td_thumbnail_title\">\n" + " " + result[i].name + "<br>$" +
result[i].regularPrice + "\n" +
```

```
                    "</td></tr></table></div> \n\n";
                }
                document.getElementById("demo").innerHTML = x;
            }
        };
        url = "https://api.bestbuy.com/v1/
products(customerReviewAverage%3E=4&regularPrice%3E150&regularPrice%3C250&(categor
yPath.id=abcat0401000))?
apiKey=xxxxxx&sort=name.asc&show=manufacturer,name,image,url,regularPrice&facet=re
gularPrice,50&pageSize=100&format=json";
        xmlhttp.open("GET", url, true);
        xmlhttp.send();
    </script>
</body>
</html>
```



Parameters in HTTP Request.

BB API allows to send request only with certain relatively restricted set of parameters. Currently we can only request products defined by: Category, CustomerReview, Manufacturer, Color and price. The formal request to BB Database then can be build only with variety of these parameters.

We will use basic HTTP Get method:

http://localhost/neurons/bestbuy/gallery.htm?
color=silver&manufacturer=apple&customerReviewAverage=3&price=1500

Our request page (gallery.htm) have basic arguments parser and few logical adjustments.

It is almost impossible in informal request define price as an exact number, so we will convert the price into the range and if no exact criteria provided, the price for request will be calculated as:

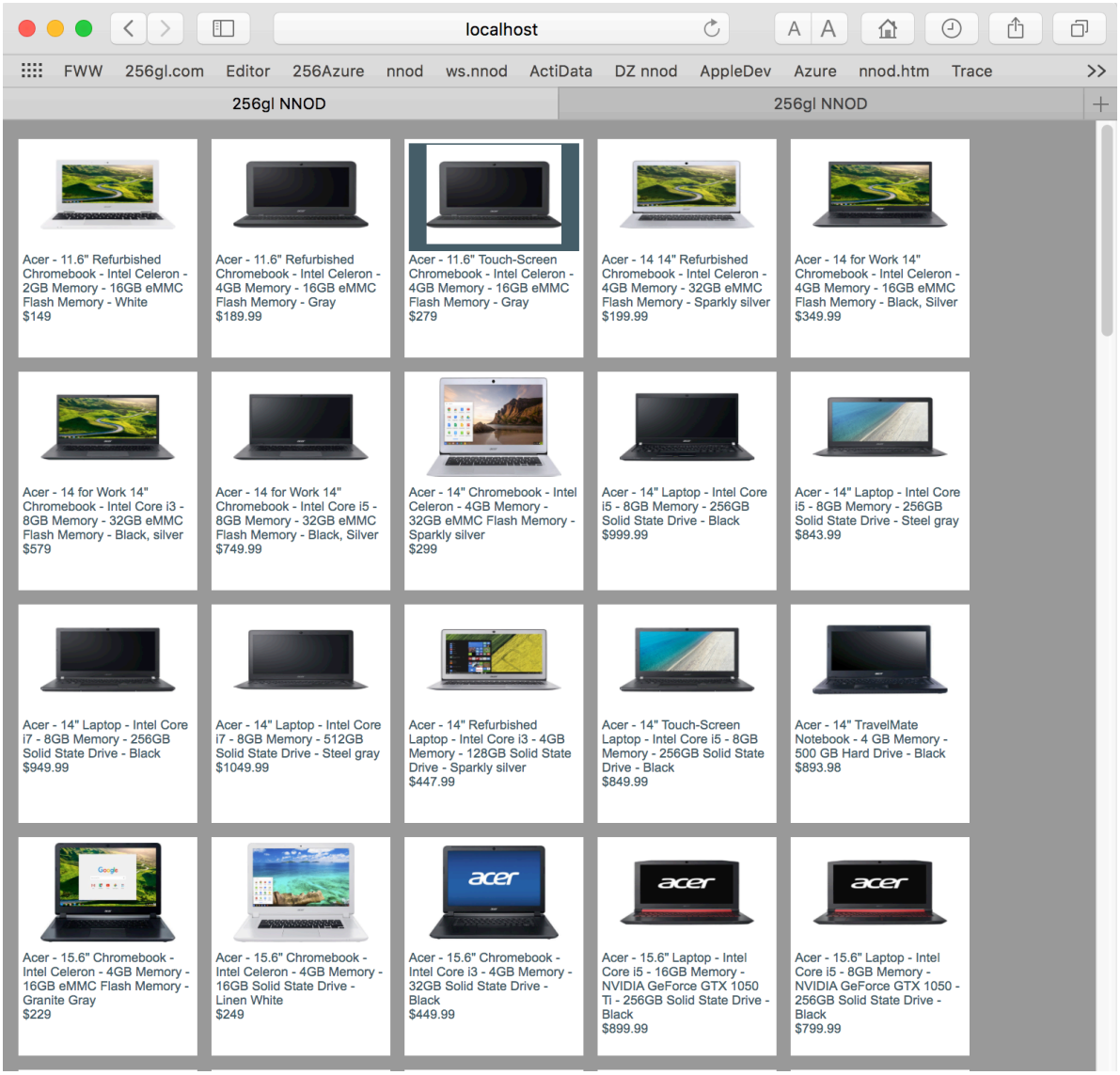
```
regularPrice_gt = price - Math.round(price / 4)
regularPrice_lt = price + Math.round(price / 4)

price >= regularPrice_gt && price <= regularPrice_lt
```

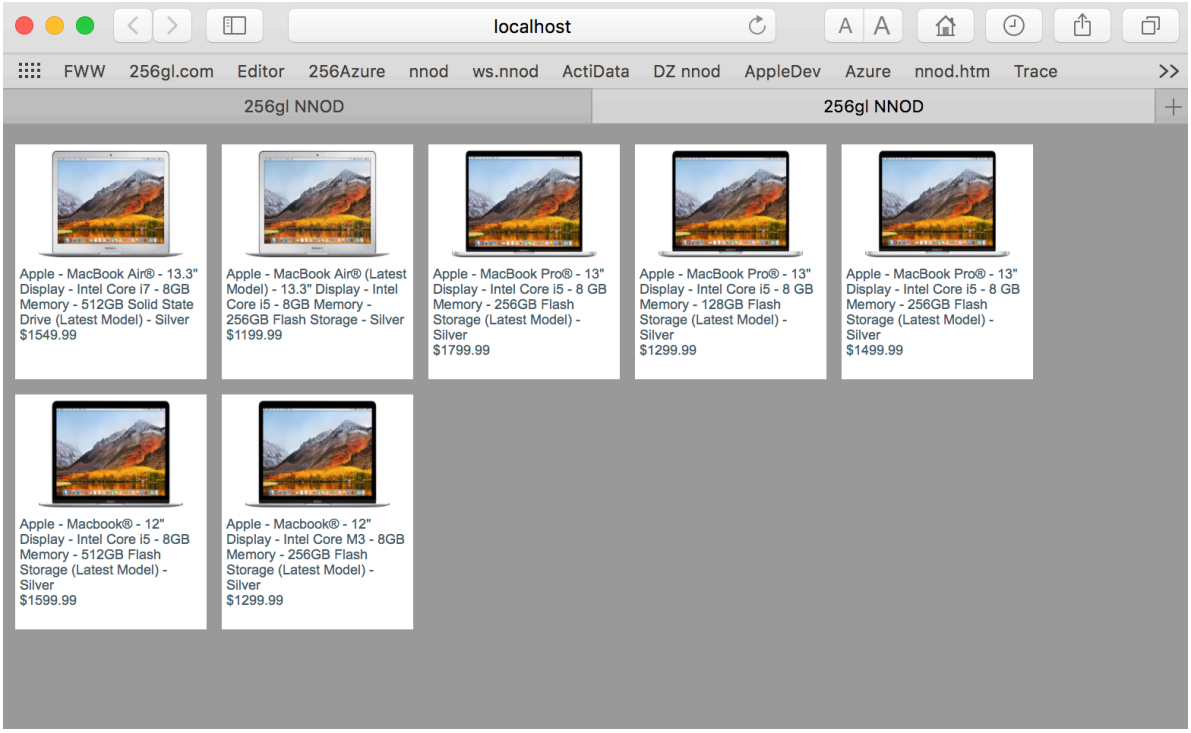
The whole arguments parsing for the page is following:

```
var api_key = "7ksBhjryZ5hxofJSEk2VBO7u";
var baseURL = "https://api.bestbuy.com/v1/products";
var categoryId = get_arguments["categoryId"];
if (categoryId === undefined)  categoryId = "abcat0502000";

//        abcat0502000 - Laptops
//        abcat0501000 - Computers
//        abcat0401000 - Cameras
//        pcmcat209400050001 - phones
//        abcat0204000 - Hesad phones
//        pcmcat241600050001 - Home Audio
//        pcmcat254000050002 - Home Automation
//        pcmcat209000050006 - iPads
//        abcat0904000 - Ovens
//        abcat0901000 - Refrigirators
//        abcat0101000 - TVs
//        abcat0910000 - Washers
//        pcmcat310200050004 -Speakers

var customerReviewAverage = get_arguments["customerReviewAverage"];
```

```
if (customerReviewAverage === undefined)
{
      customerReviewAverage = "";
}else
{
      customerReviewAverage = customerReviewAverage - 0.1
      customerReviewAverage = "&customerReviewAverage>=" + customerReviewAverage;
}
var manufacturer = get_arguments["manufacturer"];
if (manufacturer === undefined)
{
      manufacturer = "";
}else
{
      manufacturer = "&manufacturer=" + manufacturer;
}
var color = get_arguments["color"];
if (color === undefined)
{
      color = "";
}else
{
      color = "&color=" + color;
}
var regularPrice_gt = 0;
var regularPrice_lt = 100000;
var price = get_arguments["price"];
if (price === undefined || price === "")
{
      price = "regularPrice>=" + regularPrice_gt +
            "&regularPrice<=" + regularPrice_lt;
}else
{
      regularPrice_gt = Math.round(price) - Math.round(price / 4);
      regularPrice_lt = Math.round(price) + Math.round(price / 4);
      price = "regularPrice>=" + regularPrice_gt +
            "&regularPrice<=" + regularPrice_lt;

}
var url = baseURL + "(" + price + customerReviewAverage + manufacturer + color +
"&(categoryPath.id=" + categoryId + "))" +
"?apiKey=" + api_key +
"&sort=name.asc&show=manufacturer,name,image,url,regularPrice&facet=regularPrice,
50&pageSize=100&format=json";
```

With this update we can now request different category of products and update search criteria.


localhost
FWW 256gl.com Editor 256Azure nnod ws.nnod ActiData DZ nnod AppleDev Azure nnod.htm Trace
256gl NNOD
256gl NNOD
Acer - 11.6" Refurbished Chromebook - Intel Celeron - 2GB Memory - 16GB eMMC Flash Memory - White $149
Acer - 11.6" Refurbished Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Gray $189.99
Acer - 11.6" Touch-Screen Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Gray $279
Acer - 14 14" Refurbished Chromebook - Intel Celeron - 4GB Memory - 32GB eMMC Flash Memory - Sparkly silver $199.99
Acer - 14 for Work 14" Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Black, Silver $349.99
Acer - 14 for Work 14" Chromebook - Intel Core i3 - 8GB Memory - 32GB eMMC Flash Memory - Black, silver $579
Acer - 14 for Work 14" Chromebook - Intel Core i5 - 8GB Memory - 32GB eMMC Flash Memory - Black, Silver $749.99
Acer - 14" Chromebook - Intel Celeron - 4GB Memory - 32GB eMMC Flash Memory - Sparkly silver $299
Acer - 14" Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Black $999.99
Acer - 14" Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Steel gray $843.99
Acer - 14" Laptop - Intel Core i7 - 8GB Memory - 256GB Solid State Drive - Black $949.99
Acer - 14" Laptop - Intel Core i7 - 8GB Memory - 512GB Solid State Drive - Steel gray $1049.99
Acer - 14" Refurbished Laptop - Intel Core i3 - 4GB Memory - 128GB Solid State Drive - Sparkly silver $447.99
Acer - 14" Touch-Screen Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Black $849.99
Acer - 14" TravelMate Notebook - 4 GB Memory - 500 GB Hard Drive - Black $893.98
Acer - 15.6" Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Granite Gray $229
Acer - 15.6" Chromebook - Intel Celeron - 4GB Memory - 16GB Solid State Drive - Linen White $249
Acer - 15.6" Chromebook - Intel Core i3 - 4GB Memory - 32GB Solid State Drive - Black $449.99
Acer - 15.6" Laptop - Intel Core i5 - 16GB Memory - NVIDIA GeForce GTX 1050 Ti - 256GB Solid State Drive - Black $899.99
Acer - 15.6" Laptop - Intel Core i5 - 8GB Memory - NVIDIA GeForce GTX 1050 - 256GB Solid State Drive - Black $799.99


localhost
FWW 256gl.com Editor 256Azure nnod ws.nnod ActiData DZ nnod AppleDev Azure nnod.htm Trace
256gl NNOD
256gl NNOD
Apple - MacBook Air® - 13.3" Display - Intel Core i7 - 8GB Memory - 512GB Solid State Drive (Latest Model) - Silver $1549.99
Apple - MacBook Air® (Latest Model) - 13.3" Display - Intel Core i5 - 8GB Memory - 256GB Flash Storage - Silver $1199.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 256GB Flash Storage (Latest Model) - Silver $1799.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 128GB Flash Storage (Latest Model) - Silver $1299.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 256GB Flash Storage (Latest Model) - Silver $1499.99
Apple - Macbook® - 12" Display - Intel Core i5 - 8GB Memory - 512GB Flash Storage (Latest Model) - Silver $1599.99
Apple - Macbook® - 12" Display - Intel Core M3 - 8GB Memory - 256GB Flash Storage (Latest Model) - Silver $1299.99

Day 3

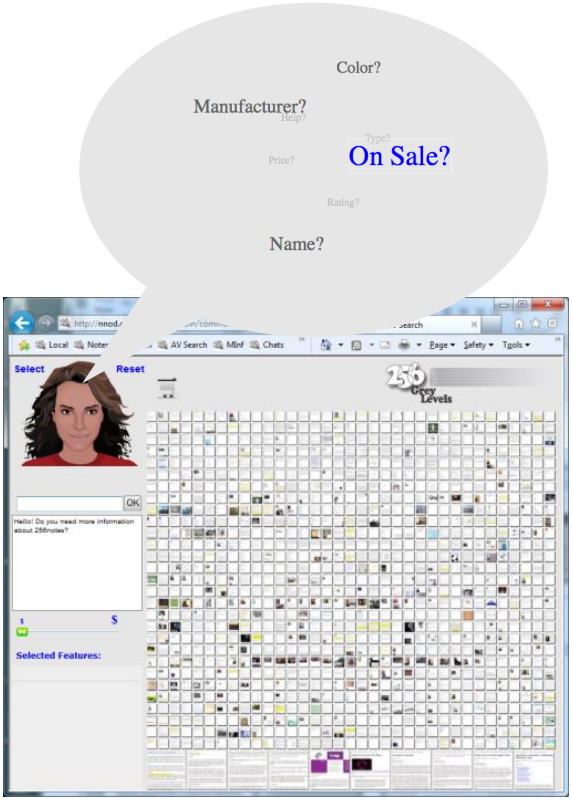
Domain of Definition, Map Informal Requests into Formal Queries.

One of the most important step in ChatBot design is clear understanding of "Domain of a Function". Informal human request will be mapped into the formal query. The goal of our design is to map practically endless variety of informal request into the limited set of allowed parameters in query, which will result with limited number of items from the DataBase. In our case, Best Buy API limited by **Product, Stores, Categories** and **Recommendation** types of queries. Understanding of this limitation makes development significantly simpler and formal.

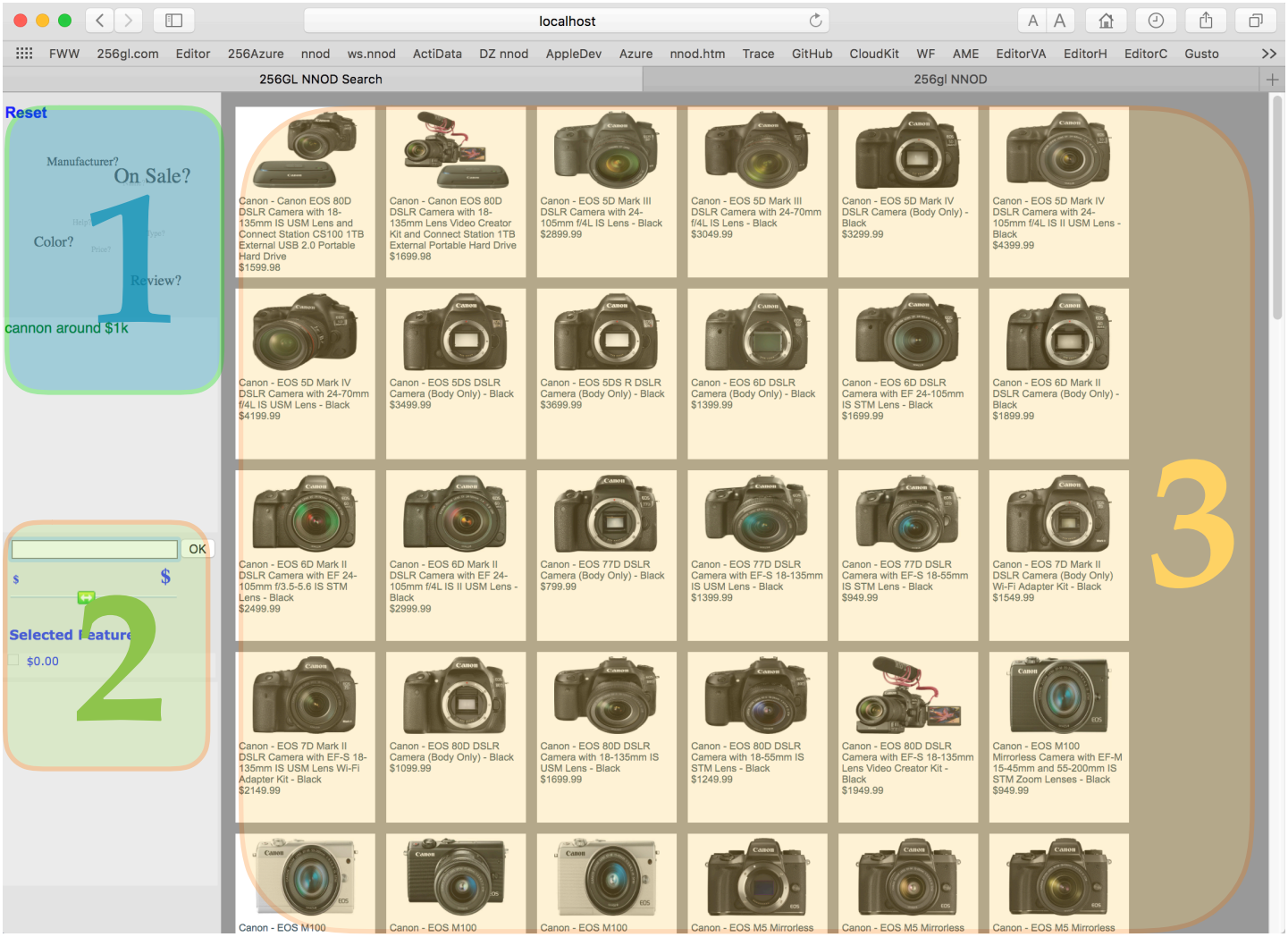
We will limit this basic ChatBot with **Product** only search but all possibilities to extend this solution with minimum modifications in the code could be dome in next version.

Do not pretend!

One of the biggest problem of modern bots - they are pretending. Huge marketing and mass media "propaganda" of AI and Turing test leads developers to the wrong design, when bots are build on the promise to imitate human. When user meet such a bot, the high exceptions leads to many unreasonable attempts to ask questions which bots unable to answer and instead of using the bot as a tool to improve search, in many cases visitors trying



to test it. For this reason, the goal is to make clear the limit of my bot. To show, that its functionality is similar to “receptionist” and it helps is limited but as in many cases “receptionist” do - they save a lot of time for a visitor and for an expert who may be the next connection in the process of finding the best solution.

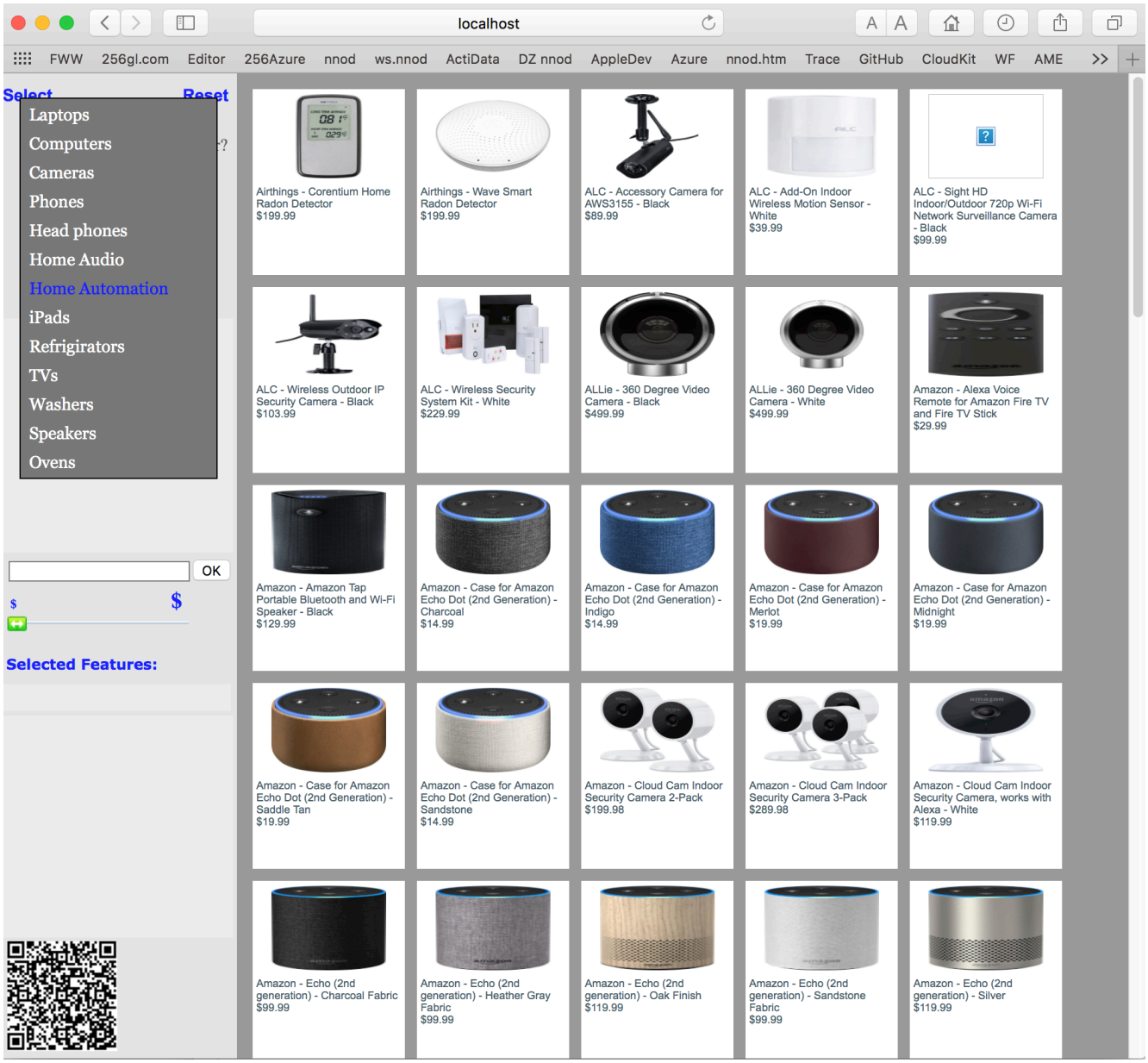


That’s why the area 1 (cloud of tags) is the model of “chatbot’s mind”. When and if the visitor will understand that the bot is limited to help make the search request formal and bot “knows” only limited number of parameters (which in many cases unknown in formal way to the visitor), the function of this bot will be clear as well as the process of search will be effective.

And to show its limit and capacity for user to avoid confusion we combined the content layer (3) together with control layers (1 and 2).

The new page combined three components: 1 - show available parameters for search, 2 - dialog and state of current interaction, 3 - content.

We also added for the future extend Selector for product categories:

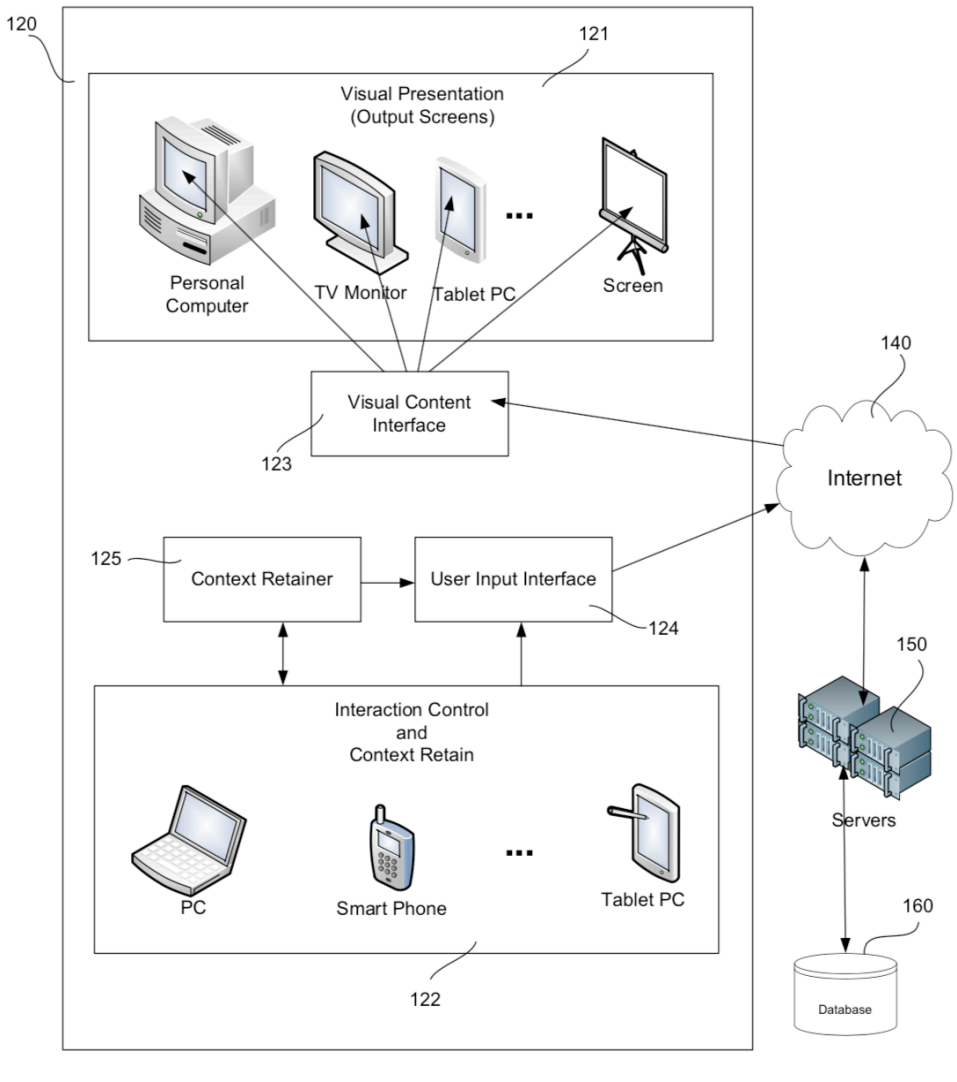


Each selector will require its own linguistic triggers.

Day 4

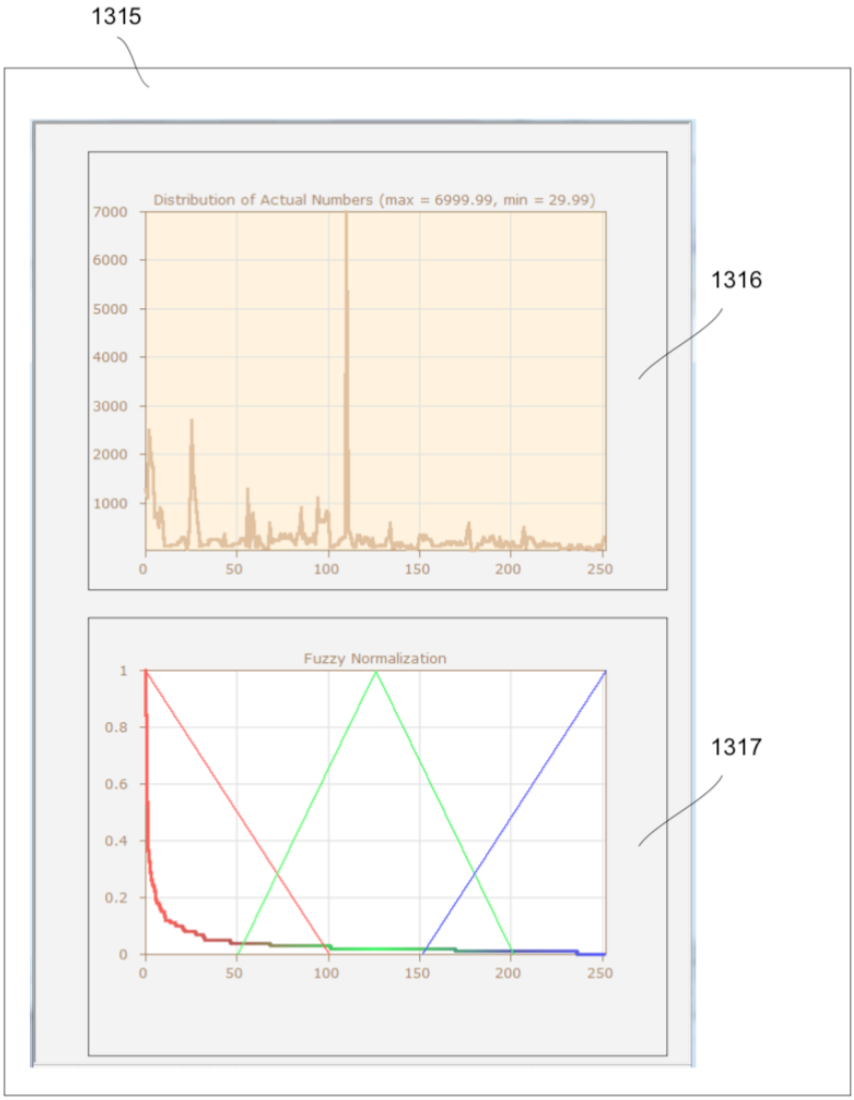
Architecture, Integration and Multi Channel Chat

One of the goal of this project is to build not a single ChatBot, but smart communication channel, capable to interact with various data sources (in our case BB API) using different devices: PC with mouse and possible touch screen, smart phones, TV, smart home devices like Alexa. Various scenarios includes comprehensive interactive search from PC, relaxed shopping in front of TV, making shopping notes using Alexa or iPhone, and helping buyer inside the store with navigation and conversation with human expert:



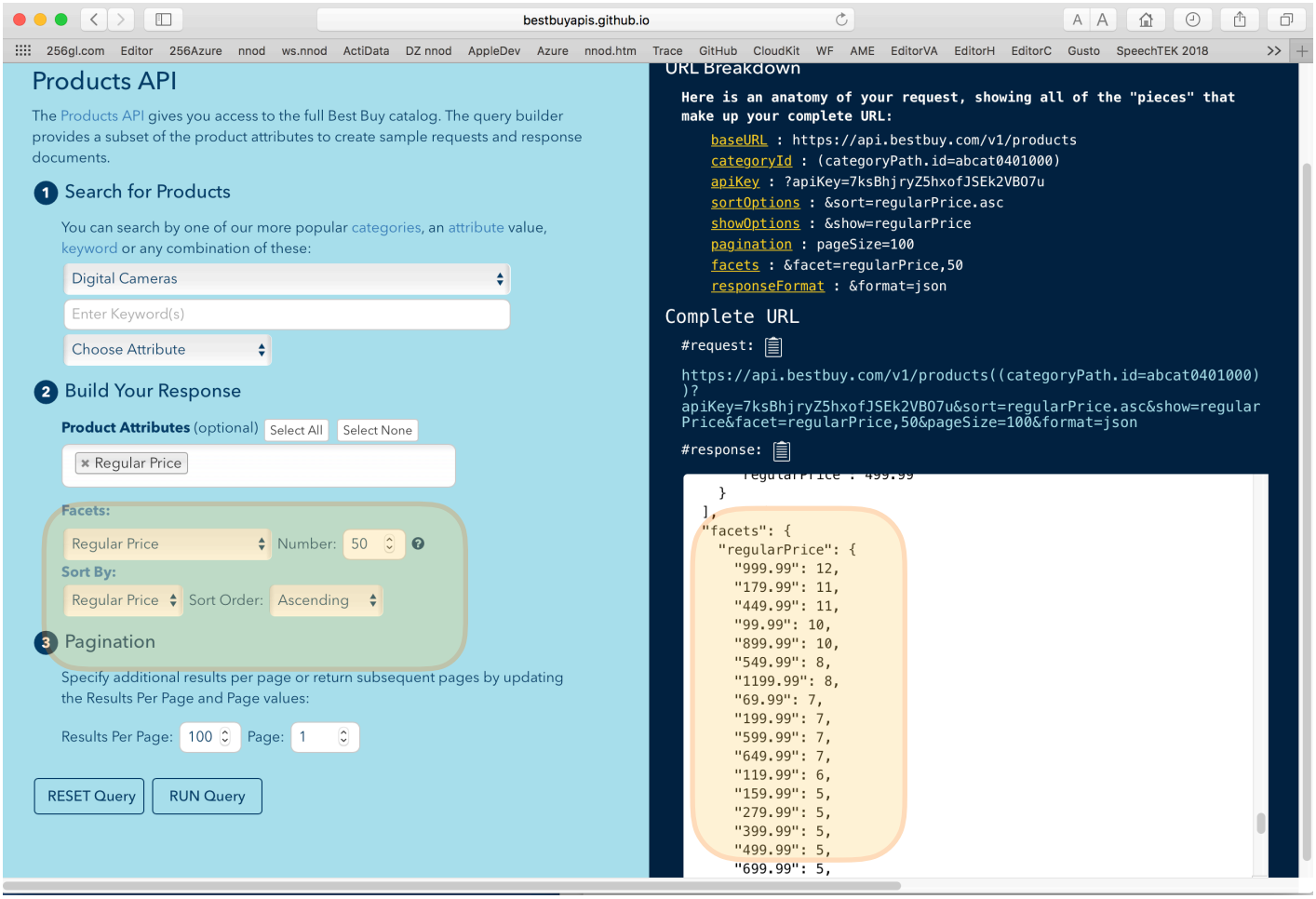
Price is one of the most often used criteria in shopping selection. Making price parameter flexible in multi channel search models is one of the most important part of the whole development. In "natural" conversation we most often using fuzzy form: "around one hundred", "my upper limit is $1000" etc. To solve this problem we are going to use the Fuzzy Set theory introduced by Lotfi Zadeh in his Fuzzy Logic and specifically in the "The concept of a linguistic variable". Zadeh's model allows us relatively easy calculate the numeric analog of linguistic terms. The model could be very simple initially, but because it based on solid formalization, the model can grow as needed.

This model allows to convert natural language terms like "average", "cheap", "best", etc. into functions and using these functions in selection requests, the result could be obtained based on traditional mathematical equations.



Desktop version of our solution may use the slider to setup the price value based on intuitive perception of “more” or “less” (left or right) position of slider.

BB API have the response parameter called Facet:



Facets may include the list of prices with their frequency. This list we use for calculating the price range and distribution for particular selection of items from BB Database.

Simple sorting and delta assigned to the current slider position will let us with minimum user’s actions select the product in price area we call “around”

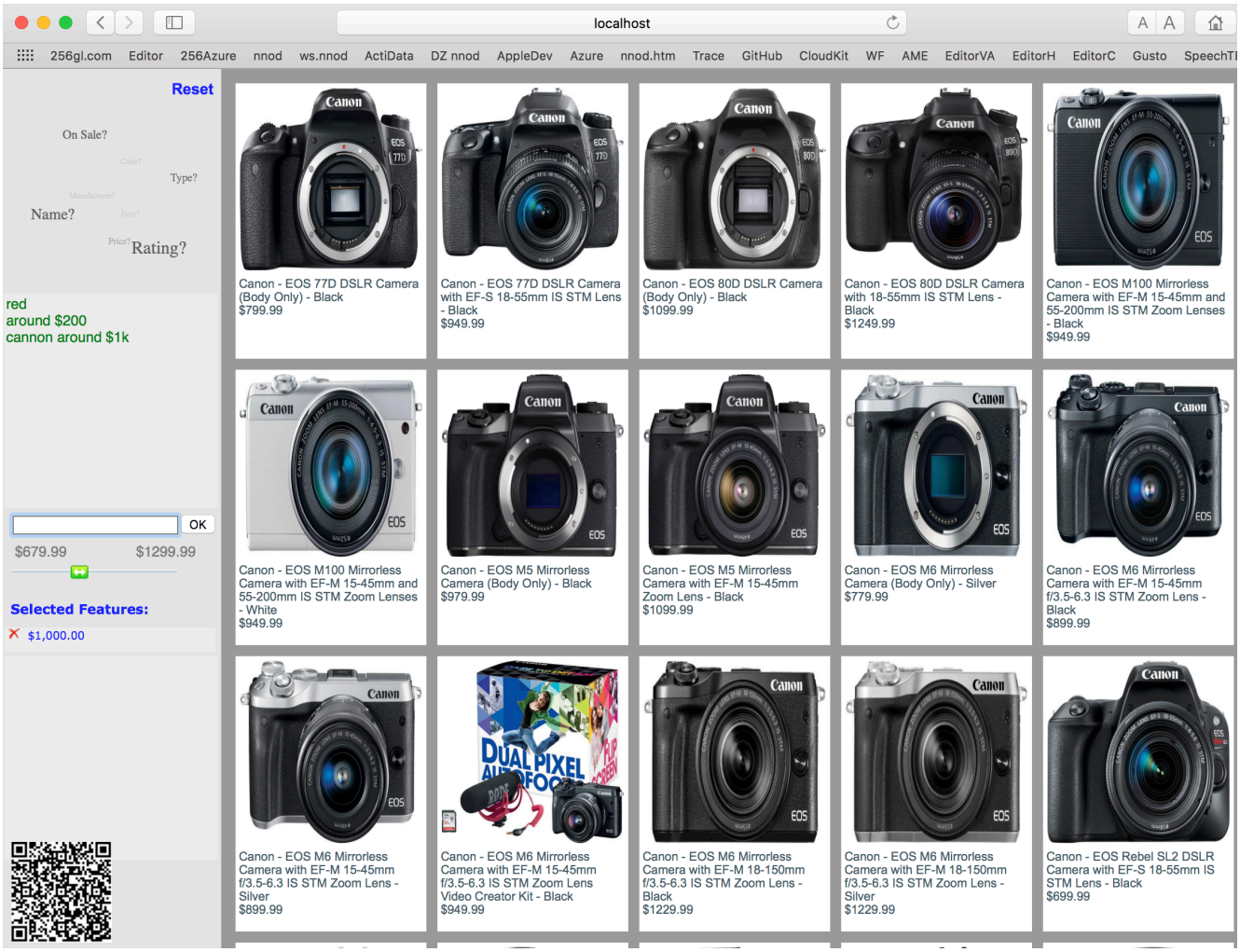
```
price = parseInt(price)
regularPrice_gt = price - price / 3;
regularPrice_lt = price + price / 3;
price = "regularPrice>=" + regularPrice_gt + "&regularPrice<="
                    + regularPrice_lt;
```

At the same time we prepared the list of all categories which currently labeled in BB Databased with formal ID and short linguistic definition:

---

abcat0502000 - Laptops
abcat0501000 - Computers
abcat0401000 - Cameras & Camcorders
pcmcat209400050001 - phones
abcat0204000 - Hesad phones
pcmcat241600050001 - Home Audio
pcmcat254000050002 - Home Automation
pcmcat209000050006 - iPads
abcat0904000 - Ovens
abcat0901000 - Refrigirators
abcat0101000 - TVs
abcat0910000 - Washers
pcmcat310200050004 - Speakers
pcmcat138100050018 - Geek Squad
pcmcat748300678080 - Instant Print Camera
pcmcat748302046986 - Instant Print Camera Cases
pcmcat748302046977 - Instant Print Accessories
abcat0410000 - Digital Camera Accessories
pcmcat128500050004 - Name Brands
pcmcat748301598672 - Wireless Doorbell Cameras
pcmcat254000050002 - Smart Home
pcmcat308100050020 - Security Cameras & Surveillance
pcmcat748300682207 - Trail Cameras & Rangefinders
abcat0409000 - Binoculars, Telescopes & Optics
pcmcat748300678486 - 360 Degree Cameras
pcmcat748300670708 - Pet Cameras
pcmcat748300670181 - Pet Supplies & Technology
pcmcat369900050003 - Drones without Cameras
pcmcat369900050002 - Drones with Cameras
pcmcat252700050006 - Drones, Toys & Collectibles
pcmcat351400050014 - Mirrorless Camera Package Deals
pcmcat214000050005 - Mirrorless Camera
pcmcat340500050007 - Security Camera Systems
pcmcat330300050003 - Dash Cameras
abcat0300000 - Car Electronics & GPS
pcmcat330300050001 - Car Safety & Convenience
pcmcat330300050002 - Back-Up Cameras
pcmcat328900050015 - Camera Mounts, Grips & Stabilizers
pcmcat328900050011 - Camera Belts & Hip Packs
pcmcat328900050008 - Universal Camera Bags & Cases
pcmcat324200050004 - All Point & Shoot Cameras
abcat0401001 - Point & Shoot Cameras
pcmcat284800050007 - Camera Batteries
pcmcat226400050024 - Camera & Camcorder Services
pcmcat171900050007 - All Camera Lenses
abcat0410018 - Camera Lenses
abcat0401005 - Digital SLR Cameras
pcmcat180000050013 - DSLR Body Only
pcmcat180400050000 - DSLR Body & Lens
pcmcat233000050008 - DSLR Lens
pcmcat258000050013 - Disposable Cameras

This pairs will be used in the next step, when we will add neural networks with linguistic triggers.

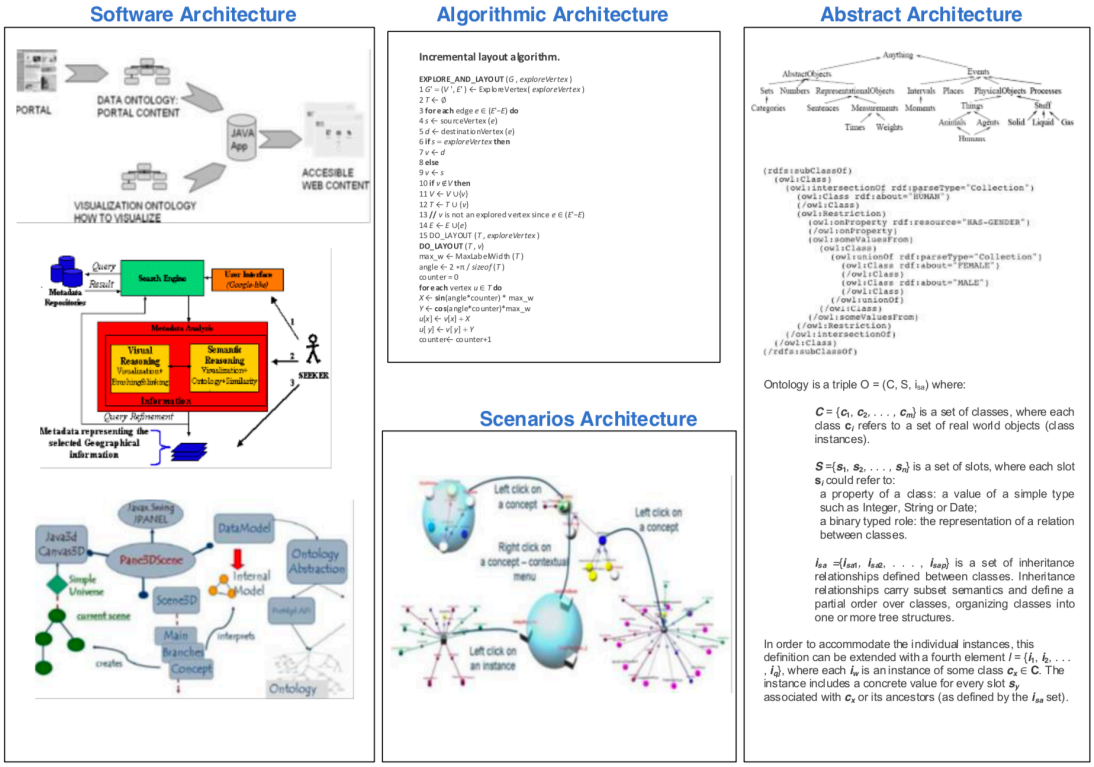


Day 5

Neural Network and Web sockets

The main goal for today is to connect solution with NNOD based triggers and add communication channel (WebSocket) for potential integration with voice (Alexa as an example) and non-PC applications running in devices like smart TV (iTV as an example.)

We will need few more technical improvements in modules organizations and callbacks and final architectural review. In a "real world" development architectural review will includes number of architectures:



In our "virtual team" all four architectures were discussed briefly and the result is following:

1. Arguments in internal calls must be kept as a "state" to support dynamic state based linguistic model.
2. Connection channel must be added as much as possible next to the presentation (gallery) layer.
3. Connection channel must support asynchronous protocol.

4. To support Alexa, which skills has 8 second timeout, the session must be identified properly and when connection potentially restored, the particular session must remember the previous conversation.

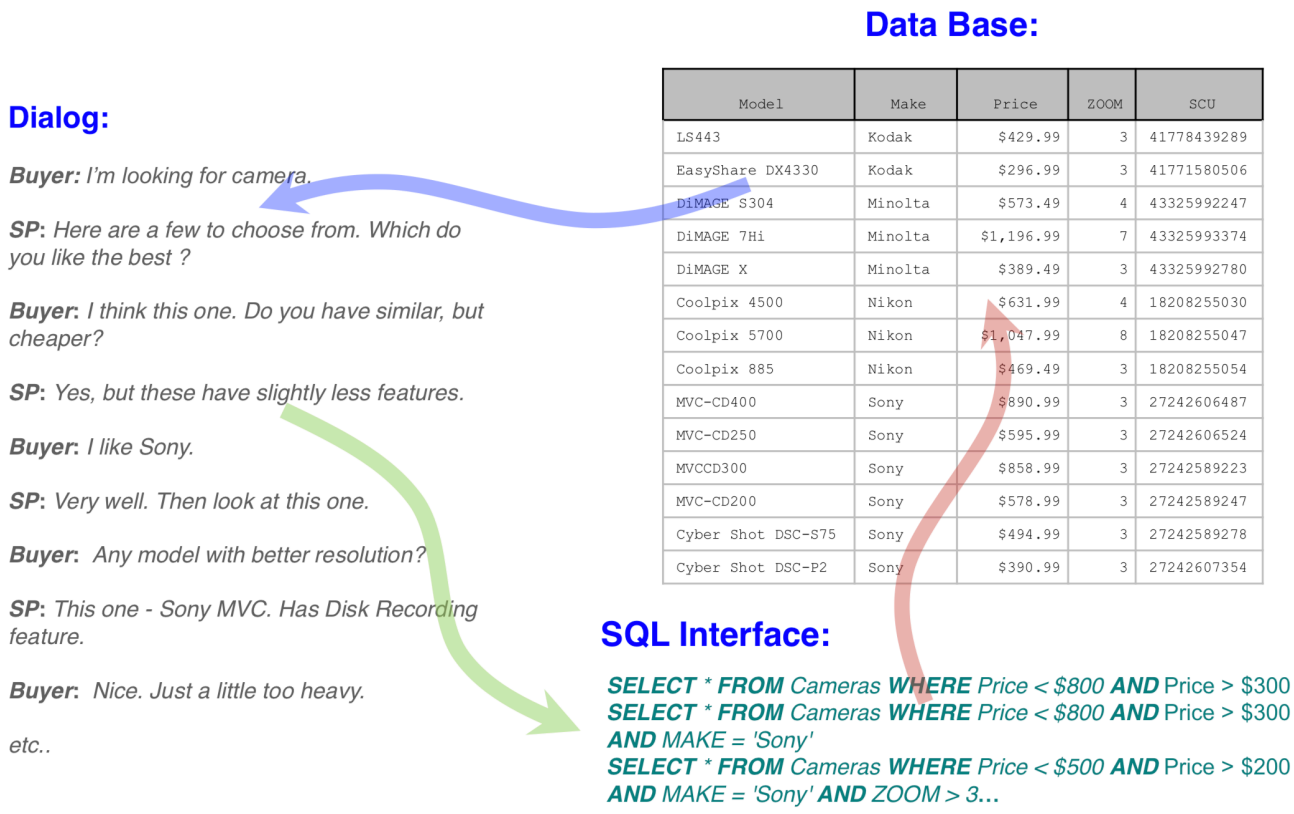
5. Mobile devices has fundamentally different screen and interaction UI, so only presentation layer could be useful.

6. One of the common scenario is - “I like something like this, but cheaper” or “better”, and this scenario could be implemented when user click or select the item, and parametric definition of this item will become the active state.

7. Cleaning parameters from the current selection is important when the search reach the “dead end” (request with particular set of parameters doesn’t return any items from the database).

8. Technical specification (Zoom, Memory, Matrix, etc.) is important and potential extension of linguistic triggers should not affect the code itself.

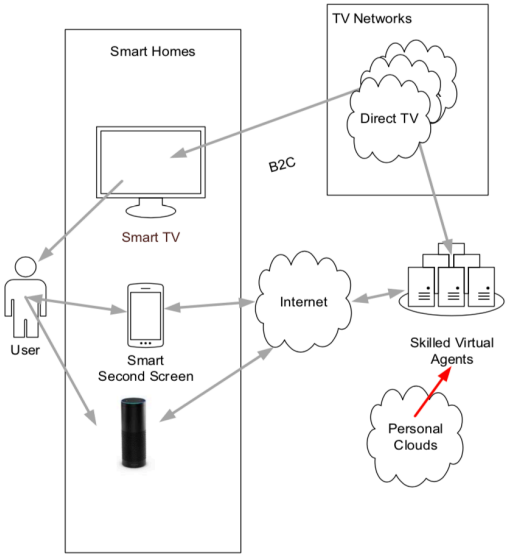
9. Software must be irrelevant to the linguistic content: if we want to make chat bot helping with a wine, very minimum adjustments should be required in basic software.



Model	Make	Price	ZOOM	SCU
LS443	Kodak	$429.99	3	41778439289
EasyShare DX4330	Kodak	$296.99	3	41771580506
DiMAGE S304	Minolta	$573.49	4	43325992247
DiMAGE 7Hi	Minolta	$1,196.99	7	43325993374
DiMAGE X	Minolta	$389.49	3	43325992780
Coolpix 4500	Nikon	$631.99	4	18208255030
Coolpix 5700	Nikon	$1,047.99	8	18208255047
Coolpix 885	Nikon	$469.49	3	18208255054
MVC-CD400	Sony	$890.99	3	27242606487
MVC-CD250	Sony	$595.99	3	27242606524
MVCCD300	Sony	$858.99	3	27242589223
MVC-CD200	Sony	$578.99	3	27242589247
Cyber Shot DSC-S75	Sony	$494.99	3	27242589278
Cyber Shot DSC-P2	Sony	$390.99	3	27242607354

There are two additional areas of using this bot in real life:

1. Similar to Audio Tour in MIA, but shopping tour in MOA
2. Home shopper - interactive second screen to ShopNBC, HSN TV, etc. channels.



Switch to TPT2

Record this episode

I like this movie

Help me to find a movie for tonight

I want to buy this

...