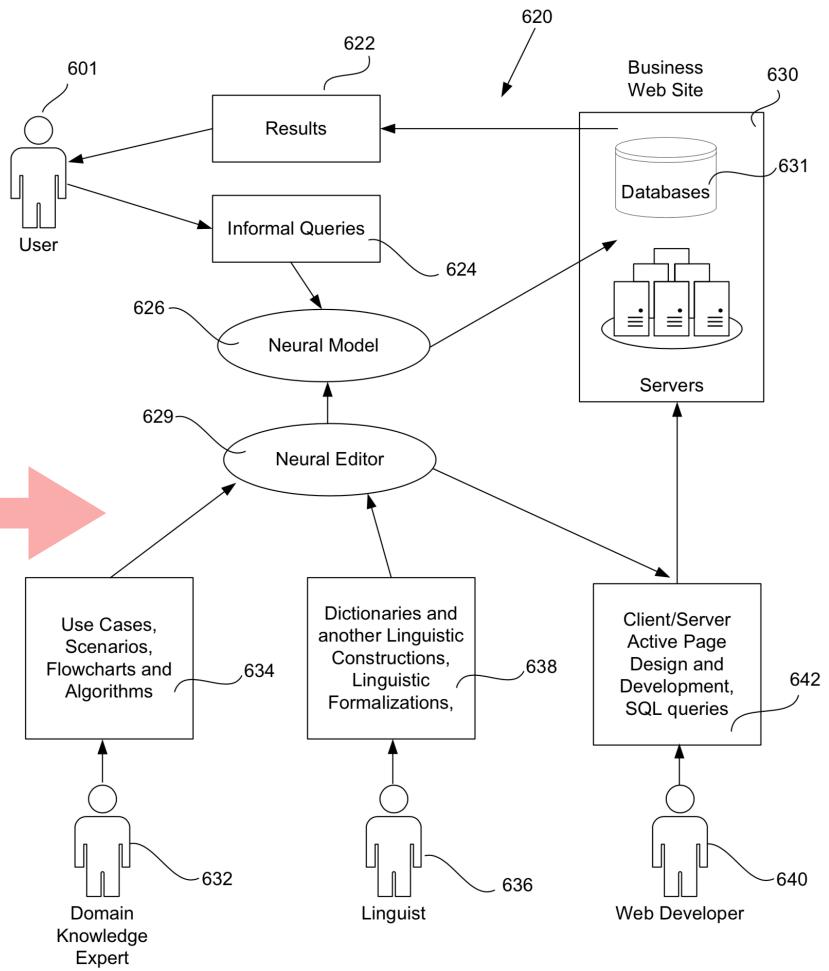
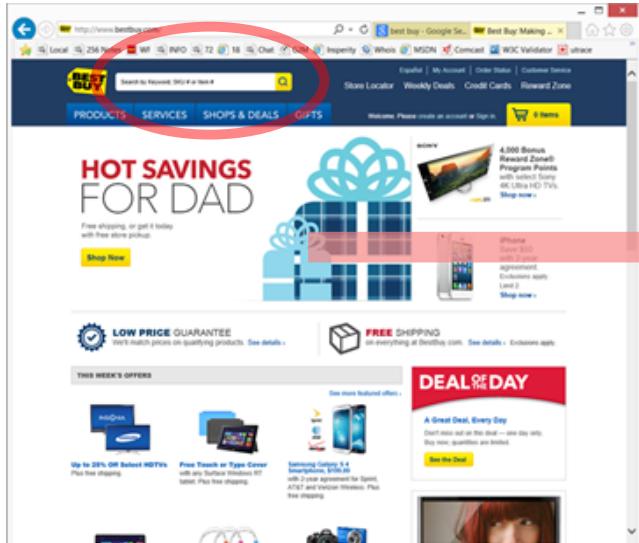


ChatBot in 7 Days

How to use Neural Networks to create Chatbot in a week

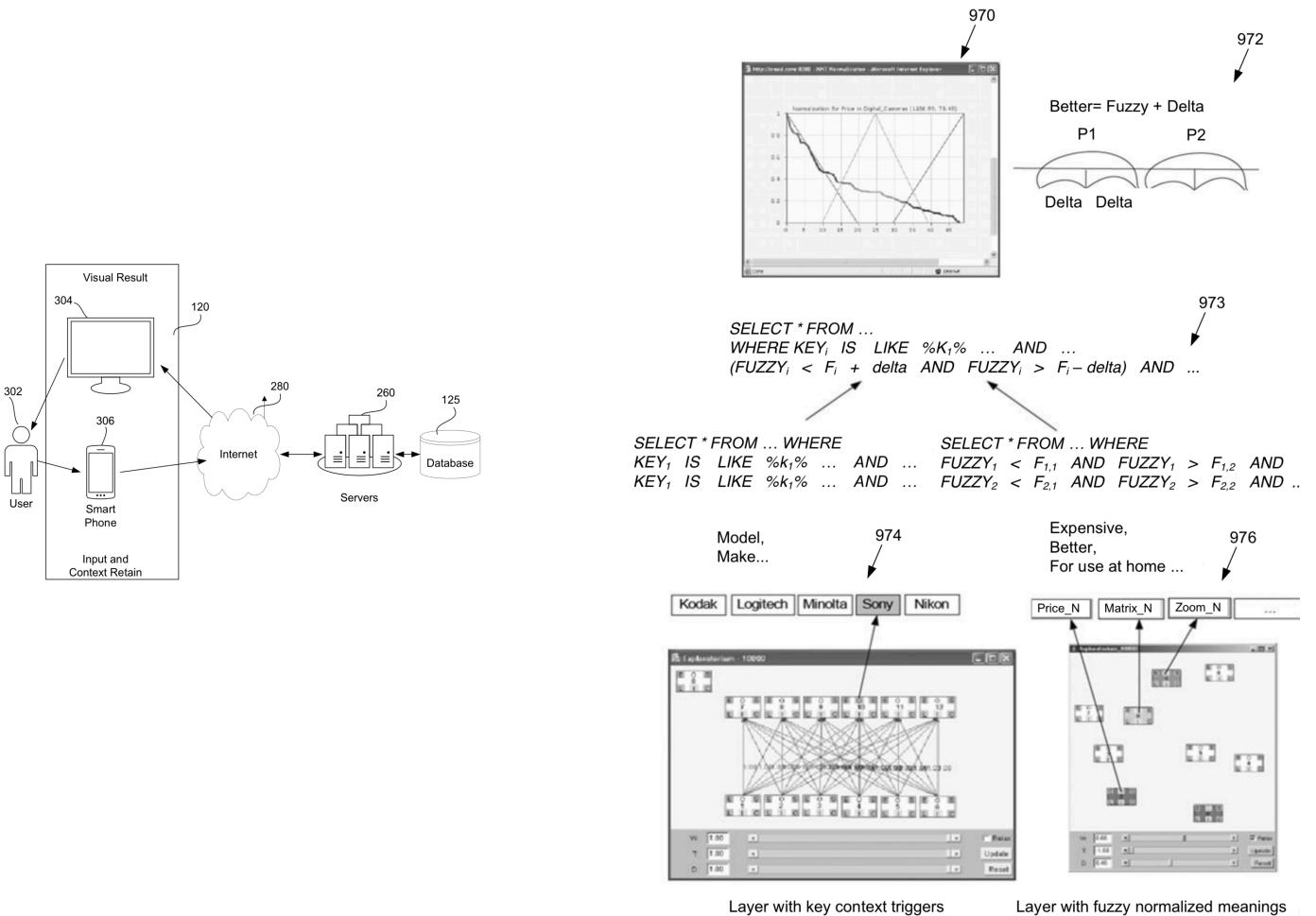
Development Diary: February 22, 2018 - February 28, 2018

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 fill application form to obtain API Key. This Key required to send requests to BestBuys database.

The next step is to learn 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:

The screenshot shows the Best Buy API Query Builder interface. On the left, there's a sidebar with navigation links: Products, Stores, Categories, Open Box, Recommendations, and Smart Lists. An API Key input field contains the value: 7ksBhjryZ5hxofJSEk2VB07u. The main area is titled "Products API".
Section 1: Search for Products. It includes a dropdown for category ("Digital Cameras"), a keyword input field ("Enter Keyword(s)"), and four range sliders for "Regular Price" (between 100 and 300), "Customer Review Average" (between 4 and 5), and two additional unlabeled sliders.
Section 2: Build Your Response. It shows "Product Attributes (optional)" with checkboxes for Name, Regular Price, Image, URL, and Customer Review Average.
Section 3: Facets. It has a slider for "Regular Price" set to 50 and a "Sort By" dropdown for "Customer Review Average" with "Descending" selected.
Section 4: Pagination. It shows "Results Per Page" set to 100 and "Page" set to 1.
On the right, the "URL Breakdown" section displays the query parameters:

```
baseURL : https://api.bestbuy.com/v1/products
categoryId : (categoryPath.id=abcat0401000)
attribute :
(regularPrice>100&regularPrice<300&customerReviewAverage>=4)
apiKey : ?apiKey=7ksBhjryZ5hxofJSEk2VB07u
sortOptions : &sort=customerReviewAverage.dsc
showOptions :
&show=name,regularPrice,image,url,customerReviewAverage
pagination : pageSize=100
facets : &facet=regularPrice,50
responseFormat : &format=json
```

The "Complete URL" section shows the generated URL:

```
#request: [copy]
https://api.bestbuy.com/v1/products(regularPrice>100&regularPrice<300&customerReviewAverage>=4(categoryPath.id=abcat0401000))?
apiKey=7ksBhjryZ5hxofJSEk2VB07u&sort=customerReviewAverage.dsc&show=name,regularPrice,image,url,customerReviewAverage&facet=regularPrice,50&pageSize=100&format=json
```

Below it, the "#response:" section shows a JSON snippet representing the API response:

```
{
  "from": 1,
  "to": 40,
  "currentPage": 1,
  "total": 40,
  "totalPages": 1,
  "queryTime": "0.123",
  "totalTime": "0.243",
  "partial": false,
  "canonicalUrl": "/v1/products(regularPrice>100&regularPrice<300&customerReviewAverage>=4(categoryPath.id=abcat0401000))?
show=name,regularPrice,image,url,customerReviewAverage&sort=customerReviewAverage.dsc&pageSize=100&format=json&facet=regularPrice,50&apiKey=7ksBhjryZ5hxofJSEk2VB07u",
  "products": [
    {
      "name": "Panasonic - LUMIX DMC-TS30 16.1-Megapixel Waterproof Digital Camera - Blue",
      "regularPrice": 179.99,
      "image": "https://img.bbstatic.com/BestBuy_US/images/products/4032/4032159_ra.jpg"
    }
  ]
}
```

At the bottom of the interface are "RESET Query" and "RUN Query" buttons.

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

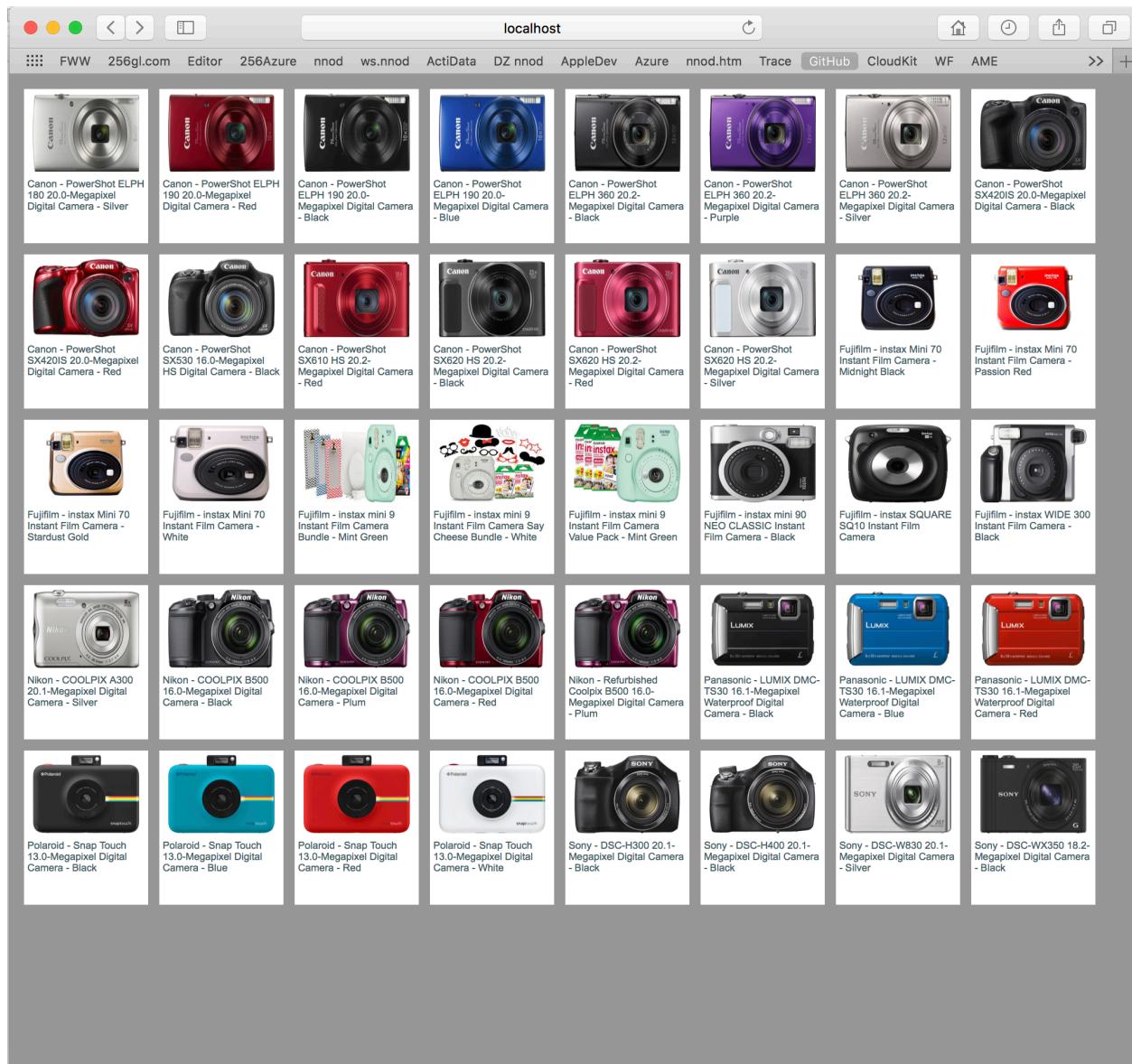
To make this solution generic as possible, we will use basic Web Server and will build interactive chat page using only standard JavaScript.

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 use PHP:



```

<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_thumtitle_". $db_counter . "\" "
class="td_thumbnail_title">\n" . " " . $obj[ 'name' ] . "\n" .
    "</td></tr></table></div> \n\n";
}
?>
</body>

```

The result:

Day 1 conclusion.

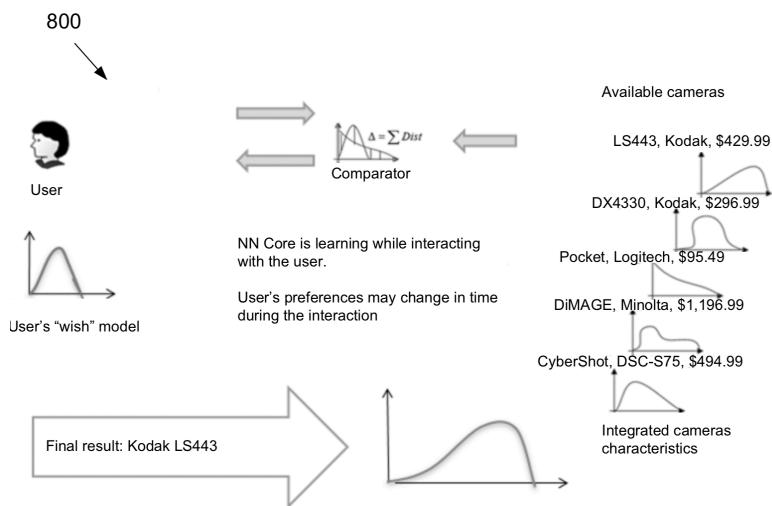
BB API Key obtained and we have full access to BB DataBase. The very first Web Page with implementation of requests to BB API built. Clear understanding of BB API now allow to start building Linguistic model for interactive search.

Day 2

Architectural/Marketing.

Value - one of the most critical questions needed to be answered before and during any development. What is the criteria we should put to measure the project advantages from similar? In our case we will use Cost of service and User's Information Demand satisfaction.

When User get an idea about buying something, he/she need to gather some information about the product. And before the very last moment when sale completed, the information for decision making will be gathered from many sources. Web, personal notes, social networks and finally expert are the network of participants in interactive process of finding the best possible item from many available in variety of sources.



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_thumbtile_" + i + "\""
class="td_thumbnail_title\">\n" + " " + result[i].name + "<br>$" +
result[i].regularPrice + "\n" +
```

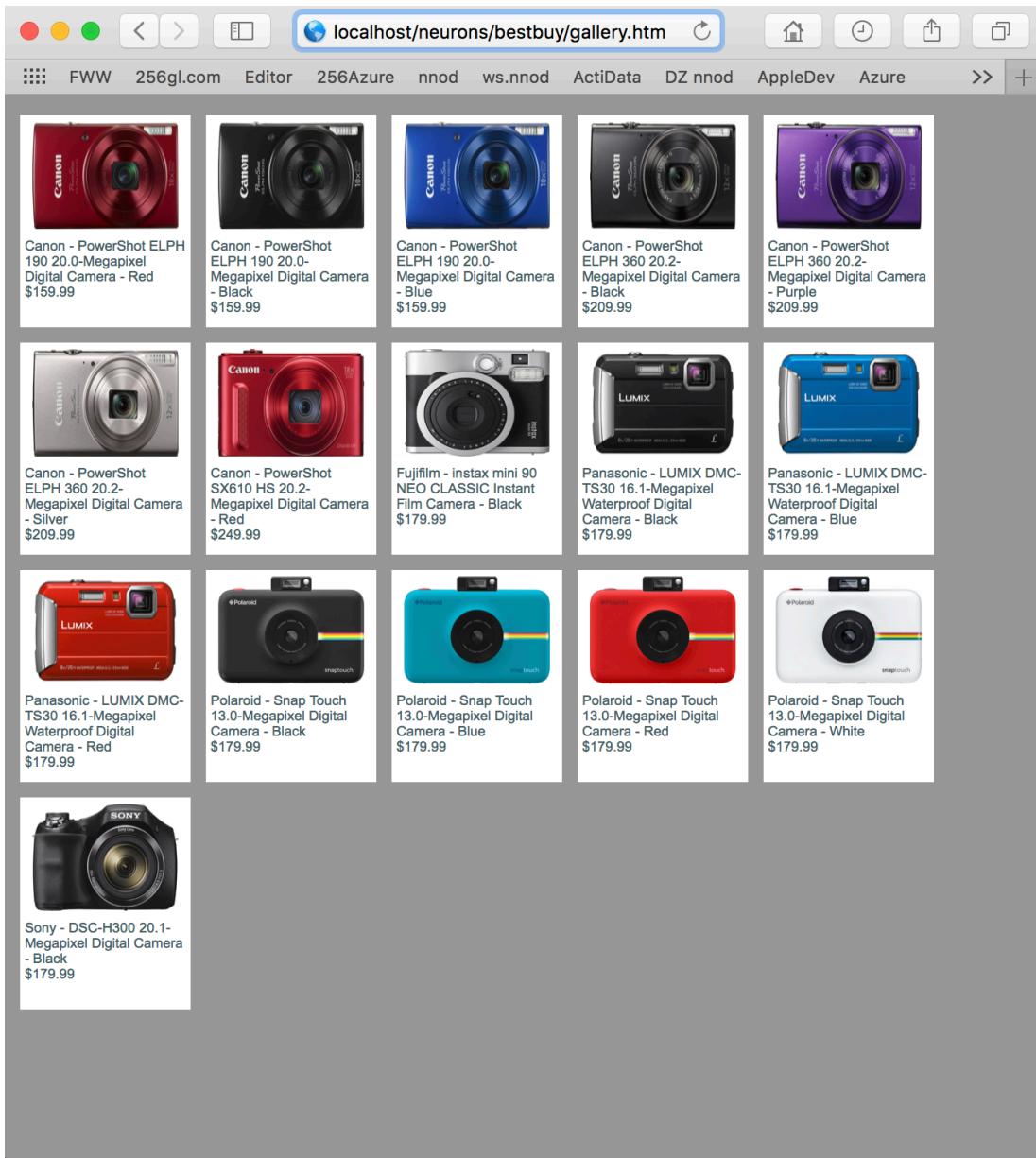
```

        "</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&\(categoryPath.id=abcat0401000\)\)?apiKey=xxxxxx&sort=name.asc&show=manufacturer,name,image,url,regularPrice&facet=regularPrice,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 = "7ksBhjryZ5hxofJSEk2VB07u";
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 - Refrigerators
//      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 - 16GB 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 done in next version.

localhost

FWW 256gl.com Editor 256Azure nnod ws.nnod ActiData DZ nnod AppleDev Azure nnod.htm Trace GitHub CloudKit WF AME EditorVA EditorH EditorC Gusto >>

256GL NNOD Search 256gl NNOD

Reset

Manufacturer? On Sale?

Help? Type?

Color? Price?

Review?

cannon around \$1k

1

2

OK

\$ \$

Selected Feature: \$0.00

3

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

It is important for ChatBot not to pretend, but instead show its limit and capacity for user to avoid confusion. For this reason we combined the “content” layer together with control layer.

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

Screenshot of a web application interface titled "localhost" showing a grid of product cards. The left sidebar contains a navigation menu with categories like Laptops, Computers, Cameras, Phones, Head phones, Home Audio, Home Automation, iPads, Refrigerators, TVs, Washers, Speakers, and Ovens. Below the menu is a search bar and a price range selector from \$ to \$\$. A QR code is also present at the bottom left.

Category	Product Image	Product Name	Price
Laptops		Airthings - Corentium Home Radon Detector	\$199.99
Computers		Airthings - Wave Smart Radon Detector	\$199.99
Cameras		ALC - Accessory Camera for AWS3155 - Black	\$89.99
Phones		ALC - Add-On Indoor Wireless Motion Sensor - White	\$39.99
Head phones		ALC - Sight HD Indoor/Outdoor 720p Wi-Fi Network Surveillance Camera - Black	\$99.99
Home Audio		ALC - Wireless Outdoor IP Security Camera - Black	\$103.99
Home Automation		ALC - Wireless Security System Kit - White	\$229.99
iPads		ALLie - 360 Degree Video Camera - Black	\$499.99
Refrigerators		ALLie - 360 Degree Video Camera - White	\$499.99
TVs		Amazon - Alexa Voice Remote for Amazon Fire TV and Fire TV Stick	\$29.99
Washers		Amazon - Amazon Tap Portable Bluetooth and Wi-Fi Speaker - Black	\$129.99
Speakers		Amazon - Case for Amazon Echo Dot (2nd Generation) - Charcoal	\$14.99
Ovens		Amazon - Case for Amazon Echo Dot (2nd Generation) - Indigo	\$14.99
		Amazon - Case for Amazon Echo Dot (2nd Generation) - Merlot	\$19.99
		Amazon - Case for Amazon Echo Dot (2nd Generation) - Midnight	\$19.99
Selected Features:		Amazon - Case for Amazon Echo Dot (2nd Generation) - Saddle Tan	\$19.99
		Amazon - Case for Amazon Echo Dot (2nd Generation) - Sandstone	\$14.99
		Amazon - Cloud Cam Indoor Security Camera 2-Pack	\$199.98
		Amazon - Cloud Cam Indoor Security Camera 3-Pack	\$289.98
		Amazon - Cloud Cam Indoor Security Camera, works with Alexa - White	\$119.99
		Amazon - Echo (2nd generation) - Charcoal Fabric	\$99.99
		Amazon - Echo (2nd generation) - Heather Gray Fabric	\$99.99
		Amazon - Echo (2nd generation) - Oak Finish	\$119.99
		Amazon - Echo (2nd generation) - Sandstone Fabric	\$99.99
		Amazon - Echo (2nd generation) - Silver	\$119.99

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:

