Journal of Computer and Communications, 2014, 2, 54-63
Published Online June 2014 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2014.28006

Scientific Research

Smart Location-Based Mobile Shopping Android Application

Günay Gültekin[1,2], Oguz Bayat[3]

[1]Electrical and Computer Engineering, Istanbul Kemerburgaz University, Istanbul, Turkey
[2]Software Developer, R & D Center, Huawei Technologies Co., Ltd., Istanbul, Turkey
[3]School of Science and Engineering, Istanbul Kemerburgaz University, Istanbul, Turkey
Email: gunay.gultekin@org.kemerburgaz.edu.tr, gunay.gultekin@huawei.com, oguz.bayat@kemerburgaz.edu.tr

Received 15 April 2014; revised 12 May 2014; accepted 10 June 2014

Abstract

In the paper, a smart location-based mobile shopping application for Android devices is proposed. The Geo-position of the user's mobile device is utilized to produce location information in shopping application (SAGO). The flow of the application is that user searches a product, and then SAGO identifies the location and searches the product on the closest electronic local stores. The idea is to get the prices from each local store with in stock information and smartly listed product list. With the proposed smart filtering algorithm, mobile shopping application achieves precise and minimum error based on searching and listing results.

Keywords

Mobile Shopping, Android Application, Unified Modeling Language (UML), Service-Oriented Modeling Framework (SOMF), Location, Price Comparison, Listing Algorithms, Filtering, Personalization

1. Introduction

Nowadays, the technology keeps improving drastically, especially on smart phones and mobile devices. Since the last decade, their hardware and software capabilities have been improved rapidly. We are able to think that the current mobile phones are as strong as computers and also they are able to compete with computers. One example is that, they have multi-core processor. It means that they are as fast as a computer, in other words, their capabilities are very broad. Therefore, the customers' requirements will increase year by year. In addition, the sales have increased tremendously in recent decades.

How to cite this paper: Gültekin, G. and Bayat, O. (2014) Smart Location-Based Mobile Shopping Android Application. *Journal of Computer and Communications*, **2**, 54-63. http://dx.doi.org/10.4236/jcc.2014.28006

As more people have more smart phones, they are more willing to use them for purchasing, searching and other purposes, instead of using a computer. The survey proves that smart phones are used as a reference to get information [1]. In addition, the big companies get benefits from mobile devices and smart phones. PayPal is experiencing an exponential growth in mobile payments and growing from $750M in 2008 to over $4B in 2011, with $20B expected in 2013 [2]. More than 3 million people have paid using the Starbucks Card Mobile application, making Starbucks the nation's largest mobile payment network. It presents in 6800 Starbucks and 1000 Target locations [3]. According to all this information, the mobile market becomes an important part of the world trade. Therefore, mobile applications are needed and produced by mobile developers.

"Online shopping" is one type of mobile application on smart phones. Furthermore, we are able to see that the web market is losing the people's focus, on the other side the focus is shifting towards the mobile market. Moreover, the people have a limited time nowadays because the technology has been improving and tons of work must be finished in a specific time; therefore the time becomes important for our daily work. In addition, the people want to decrease time consuming tasks on daily routine works by using the mobile devices. According to it, market analyst says that the people use their phones more than that in the last two decades.

In general, the people are willing to buy the cheapest products. Also, the location of shopping malls is an important factor, too. People prefer to choose closer shopping malls for buying a product even though the product's price is high. Furthermore, they prefer to buy the cheapest product in the closest shopping mall. Therefore, the people have to search the location of shopping malls and products which are available at each shopping mall. Consequently, the location information is the key value for mobile applications. The location information is used as a standard feature which is used in the other applications and also, this mobile shopping Android application's coverage is not limited. The application is easily extendable according to the users' requirements. The Android application is developed in Android 2.2 version, Application Programming Interface (API) level 8. Besides this information, the system architecture is designed by using Service-Oriented Modeling Framework (SOMF) based on Unified Modeling Language (UML) and object-oriented programming language is used in development process. Designing with UML is simplifying complexity of the system and helps us to understand the architecture of this paper project. In the development process, the users' input data which has been brought before searching products in the nearest electronic super-stores and also the names of local shopping stores are used in a "Smart Filtering algorithm". Moreover, simple filtering and cleaning technics such as Agglomerative Clustering Algorithm, Greedy Search Algorithm, and a Levenshtein distance are used in SAGO mobile shopping application. Although the mobile application is developed for android devices, it will be easily adopted and developed for iOS devices. By using this android mobile application, the users are able to search the products in closer areas without knowing the location of the shopping mall, and compare the prices that are getting from each local electronic super-store and subsequently decide where to buy. According to these, the users save both their money and time by using the android mobile application.

2. Architecture of Sago Mobile Shopping Application

The system architecture diagram was created by using SOMF depends on UML. SOMF has been proposed by author Michael Bell as "a holistic and anthropomorphic modeling language for software development that employs disciplines and a universal language to provide tactical and strategic solutions to enterprise problems" [4]. In short, SOMF is a service-oriented development life-cycle methodology.

As seen in **Figure 1**, the system architecture depends on three layers; Resource Layer, Data Access and Extraction Layer, Presentation Layer.

The top layer of the system architecture is "Resource Layer". This layer includes the related data which is used in the SAGO mobile shopping Android application. The most part of the data is collected from internet resources which are the electronic local stores in Turkey. The local electronic stores which are used in the SAGO mobile shopping Android application are; Teknosa, Darty, Bimeks, Vatan Computer, Electro World.

There are also many other local electronic stores. These are the most popular electronic stores in Turkey. In future development, the other electronic local stores will be able to be added to the SAGO mobile shopping Android application. Another part of the data is gathered from Google. Google provides a variety of data to the developers and researchers. In this paper, the two Google Web APIs are used to collect the data. The first one is Google Places API and the second one is Google Map API.

The second layer of the system architecture is "Data Access and Extraction Layer". This layer includes tools

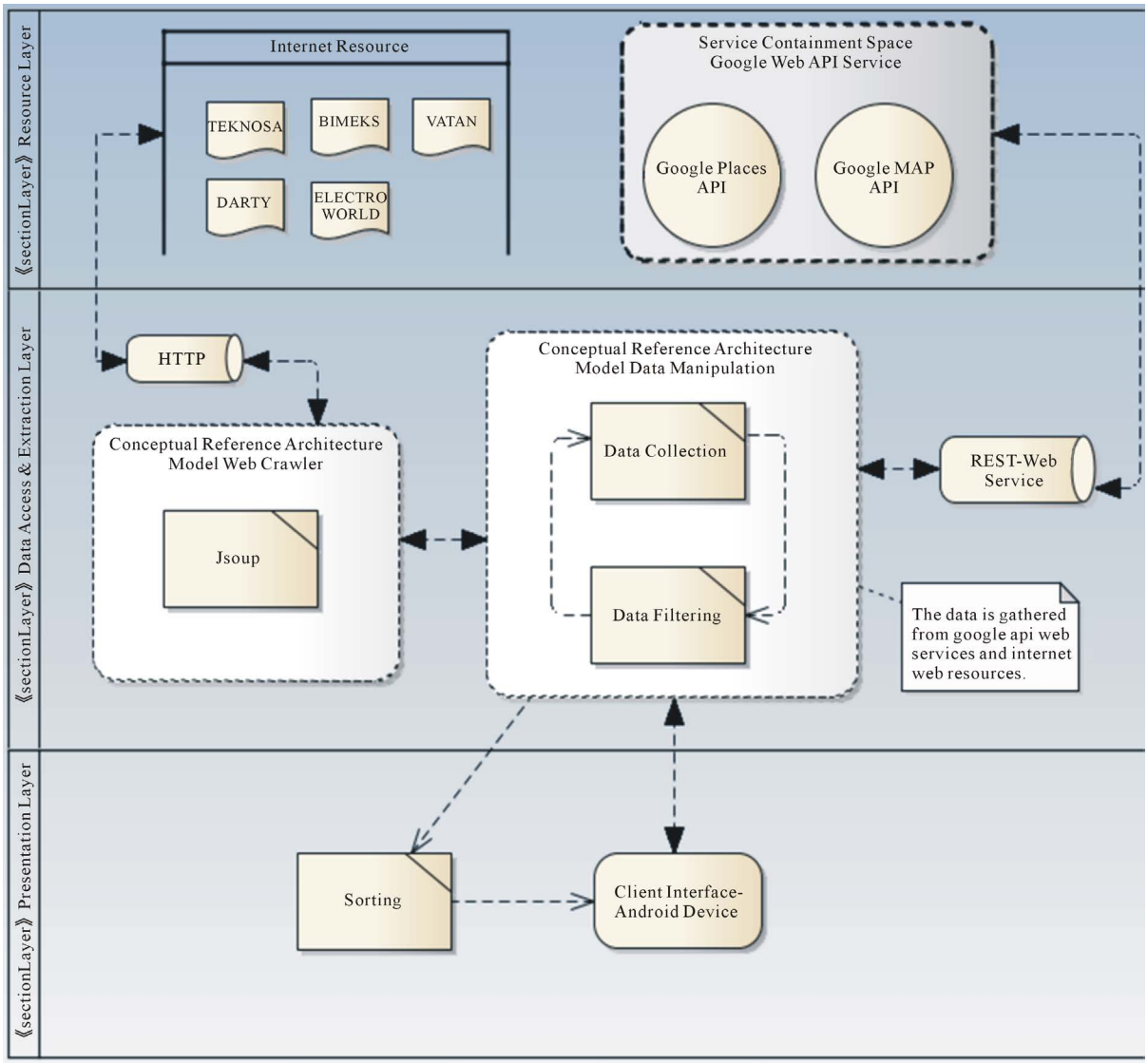

Figure 1. System Architecture of SAGO.

and data-extraction. The tool, which named as Jsoup, defined as "Jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and JQuery-like methods" [5]. Jsoup is a Java Hyper Text Markup Language (HTML) Parser and an open source project. This project distributed under the Massachusetts Institute of Technology (MIT) license. This license allows developers to use it freely in their applications. In this paper, Jsoup uses HTTP protocol to collect the data from the web resources. Moreover, the data is also gathered from Google APIs through REST Web service. In addition, after collecting from Google APIs, the filtering is needed to eliminate the unwanted and unrelated Information.

The third layer of the system architecture is "Presentation Layer". This layer includes sorting the related results that are gathered from the Data Access and Extraction Layer. Presentation layer aims to present the result in a meaningful and a logical way that covers the user requirements. The basic sorting technique is used in SAGO because the more complex sorting technique is not needed.

3. Smart Filtering Algorithm

In this paper, the Smart Filtering algorithm is a combination of different types of algorithms. Smart filtering al-

gorithm includes:

- Greedy Search Algorithm
- Agglomerative Clustering Algorithm
  - Normalization
  - Levenshtein distance
- Scoring products

Greedy Search Algorithm is used for removing unrelated information from the data. This algorithm is modified according to the SAGO mobile shopping Android application's requirements. The same algorithm and parameters in Greedy Search Algorithm are used, however, this time; "R" is the summation of products. The products come from internet resources. Moreover, the products are also filtered in that algorithm.

The Agglomerative Clustering Algorithm is more complex than Greedy Search Algorithm including some calculation and filtering. It is also used for clustering the results. These are made of product search results. These results enable the mobile user to see prices at the nearest electronic local stores in SAGO mobile shopping Android application.

In data mining, the Agglomerative Clustering Algorithm is one type of strategy for hierarchical clustering. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters in data mining [6]. In addition, the clustering is an unsupervised (does not have data for demonstrating in the learning process) learning method.

The SAGO Android mobile shopping application uses clustering approach to help the Smart Filtering algorithm to be able to show product search results to the mobile users efficiently. According to the investigation of my product search results from internet resources; for instance, the mobile user searches iPhone, different types of products are gathered and some of them are unrelated. The unrelated result should not be shown to the customer therefore the product search results are in a group by using the each result item's price because we know that the price of the iPhone mobile device is in a certain range.

The Agglomerative Clustering Algorithm or Agglomerative Hierarchical Clustering starts with objects (one-point, singleton) which are set as a cluster, in other words, each object is a cluster at the beginning. Then, the object is added to the closest pair of clusters for each of the iteration if a similar criterion is validated. This algorithm continues until all of the data is merged down as one cluster and this is an agglomerative strategy which is a "bottom up" approach. This approach is expressed more formally in **Figure 2**. Moreover, traditional hierarchical algorithms use similarity or distance/proximity matrix. The algorithm is shown in **Figure 2** and the key operation is the calculation of the distance/proximity matrix between two clusters.

There are lots of methods that calculate the distance between two clusters [7]. Euclidean distance is used for distance measurements. It is chosen because of measurements of the calculation process time.

In an Agglomerative Clustering Algorithm expressed in **Figure 3**, x and y are the input parameters of the algorithm. $x_1, x_2, x_3, \cdots \in x$, x_1 is the second input parameter of the first product and $y_1, y_2, y_3, \cdots \in y$, y_1 is the

Compute the distance/proximity matrix between the input data points, if necessary.
Let each data point be a cluster
Repeat
    Merge the closest two clusters
    Update the distance/proximity matrix
Until Only one cluster remains.

Figure 2. Basic agglomerative hierarchical clustering algorithm.

1) Remove unrelated products
2) Calculate the representative price of the product
3) Normalize the output of the step 2 (x)
4) Measure similarity score between search keyword and the product title (y)
5) Use modified agglomerative clustering algorithm
6) Compute

$$a * x + b * \left((90 - y) / 90 \right) + \text{clustering score}$$

where,

$$a > b$$

Figure 3. Smart filtering algorithm.

first input parameter of the first product. Moreover, the first iteration, the algorithm does not compute any distance, and all points behave like a cluster. The next iteration, the distance matrix is calculated and then the closest two clusters merge down to create a new single cluster. This algorithm keeps repeating until only one cluster remains.

The first parameter (x) is generated after two operations are rendered.

The first operation is clustering the similar product using their prices that were mentioned in [8]. In this paper, it is mentioned about the similar products in the same category usually have the similar prices therefore differing of between products is the price information. In this paper, to provide meaningful comparison, each actual price is converted to a representative price and the formula gives both upper bound limit and lower bound limit of the price. Their formula is shown below.

$$P^I = \begin{cases} 10, & \text{if } P \le 10 \\ \left\lceil \frac{P}{m} - 0.5 \right\rceil * m, & m = 10^{\lfloor \log P \rfloor} \text{ if } P > 10 \end{cases} \tag{1}$$

In this formula, P is the price and P^I is the representative price. Moreover m is a variable which value is 10 TL in this formula. This assumption depends on explanation in [8]. According to this research paper, approximately ten dollar difference is much more noteworthy for low-price products than for high-price products [8] so the value of m is 10 Turkish Liras. For instance, a pair of headphones is 28 TL and another similar pair of headphones is approximately 3 or 5 TL the difference therefore, 10 TL is a much more significant price on cheap products. In addition, if P is 28 TL, the representative price will be 30 TL. The representative of the price that is between 26 and 35 TL is 30 Turkish Liras.

The second operation is the normalization process. The output of the first operation in here is the input of the normalization process. After the normalization process is rendered, the first parameter (x) is generated and the price of products is scaled between 0 and 1. This is needed for presenting the products in a two dimension coordinate system and also the total calculation time is low. The normalization formula is shown below.

$$\text{Normalization}\left(X_{price}\right) = \frac{X_{price}}{\text{MaxPrice}} \tag{2}$$

In this formula, X_{price} is the input parameter and it is the representative price of the product. "MaxPrice" is the maximum price which is selected from the product result list. Therefore, this formula scaled down the price between 0 and 1.

In general, the SAGO mobile shopping android mobile application searches electronic products. In addition, it is a PoC mobile application therefore, the three products are determined which are laptop, iPhone, and camera. The aim of this formula is to sort the high price products. These products are very expensive so this formula works very well in that domain.

The second parameter is produced by a string similarity function. In my paper, the Levenshtein distance algorithm is used. The aim of this algorithm is to measure the difference between two sequences or strings. It depends on the number of changes required to change one word into the other [9].

The Levenshtein distance between two strings a, b is given by $lev_{a,b}\left(|a|,|b|\right)$ where [9],

$$lev_{a,b}\left(i,j\right) = \begin{cases} \max\left(i,j\right) & \text{if } \min\left(i,j\right) = 0, \\ \min\begin{cases} lev_{a,b}\left(i-1,j\right)+1 \\ lev_{a,b}\left(i,j-1\right)+1 \\ lev_{a,b}\left(i-1,j-1\right)+\left[a_i \ne b_j\right] \end{cases} & \text{otherwise.} \end{cases} \tag{3}$$

For instance, the Levenshtein distance between "gunay" and "gulen" is 3. The first two characters are the same. The third, fourth, fifth characters need to change as;

1) gu**n**ay → gu**l**ay (substitution of "l" for "n")
2) gul**a**y → gul**e**y (substitution of "e" for "a")
3) gule**y** → gule**n** (substitution of "n" for "y" at the end) Therefore the result is 3 again.

This distance algorithm has several constraints.

- It is always, at least the difference in the sizes of the two strings or words [9].
- It is at most the length of the longest string [9].
- It is zero if and only if the strings are equal [9].
- If the strings are the same size, the Hamming distance is an upper bound on the Levenshtein distance [9]. Hamming distance measures the minimum number of substitutions required to change one string to another [10].
- The measured distance between two strings is no greater than the sum of their Levenshtein distances from a third string [9].

To sum up, the previously mentioned algorithms are now combined and then, the Smart Filtering algorithm is made. The first step is filtering the unrelated products by using Greedy Search Algorithm.

The second step is calculating the representative price of the product by using the formula given in Equation (1). This step causes that the high price products will be seen at the top of the search result list. Therefore; in the similar products, Smart Filtering Algorithm always sorts the high price products.

The third step is the normalization process. This step must be done because of decreasing the calculation time, put it in another word, if the step is skipped, the calculation time of step 5 and step 6 will be long. This normalization process increases the product efficiency.

In the fourth step, each product search result is assigned a value according to similarity of the keyword searched which was explained in previous paragraphs. In this process, the Levenshtein distance is used to measure the similarity. Moreover, the assigned value is between "1" and "0". "1"means the product search result item is a suitable product which is expected; on the other hand, "0" means the result item is not an expected product.

The fifth and sixth step is to calculate the given function in **Figure 3** and the result is the score of the product that is a search result item. In the formula, the first calculation until addition scales the product title in a range that is between "0" and "1". "*a*" and "*b*" are the coefficients of the formula. The important key is the value of *a*. "*a*" should be greater than the value of "*b*" because the price information is an important factor to choose a product according to the customer view.

In the last step, the products are sorted according to their price in descending order. **Figure 4** shows a demonstration of the title/price range on the x-y coordinate. The figure shows clustering. In this demonstration, the search keyword is the "iPhone". After Smart Filtering is done on the search result data, the clustered products are made. In this algorithm "*a*" is 10 and "*b*" is 2.

Moreover, here is the dendrogram view of "camera" search results after the clustering algorithm renders on this data. The dendrogram simplifies the understanding of the complexity of the "hierarchical clustering" algorithm. The view shows the clusters in each level. The top level which the score of the algorithm is 24 expresses the least related search result in a given data. This score is used as negative in the Smart Filtering algorithm. As a result, the product smart list is shown to mobile user with a user friendly design. The expected products of a search result are in the upper side of the smart list. **Figure 5** shows the nearest clusters to zero contain relevant search results.

4. Conclusion and Results

This paper describes developed mobile shopping android application for mobile users that searches and lists the desired products with location information. The main idea is to get the prices from each local store with in stock information and smartly listed product list.

The efficiency of the SAGO is improved by optimizing the data gathering time. Gathering the data from Internet

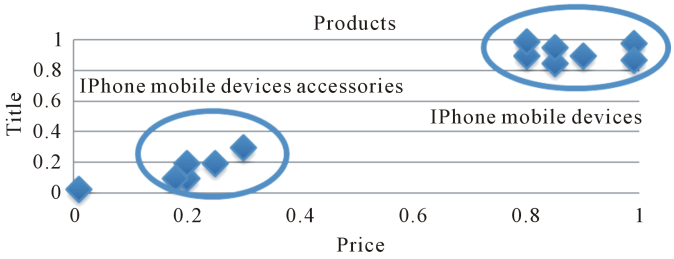

Figure 4. Sample distribution of the products in coordinate system.

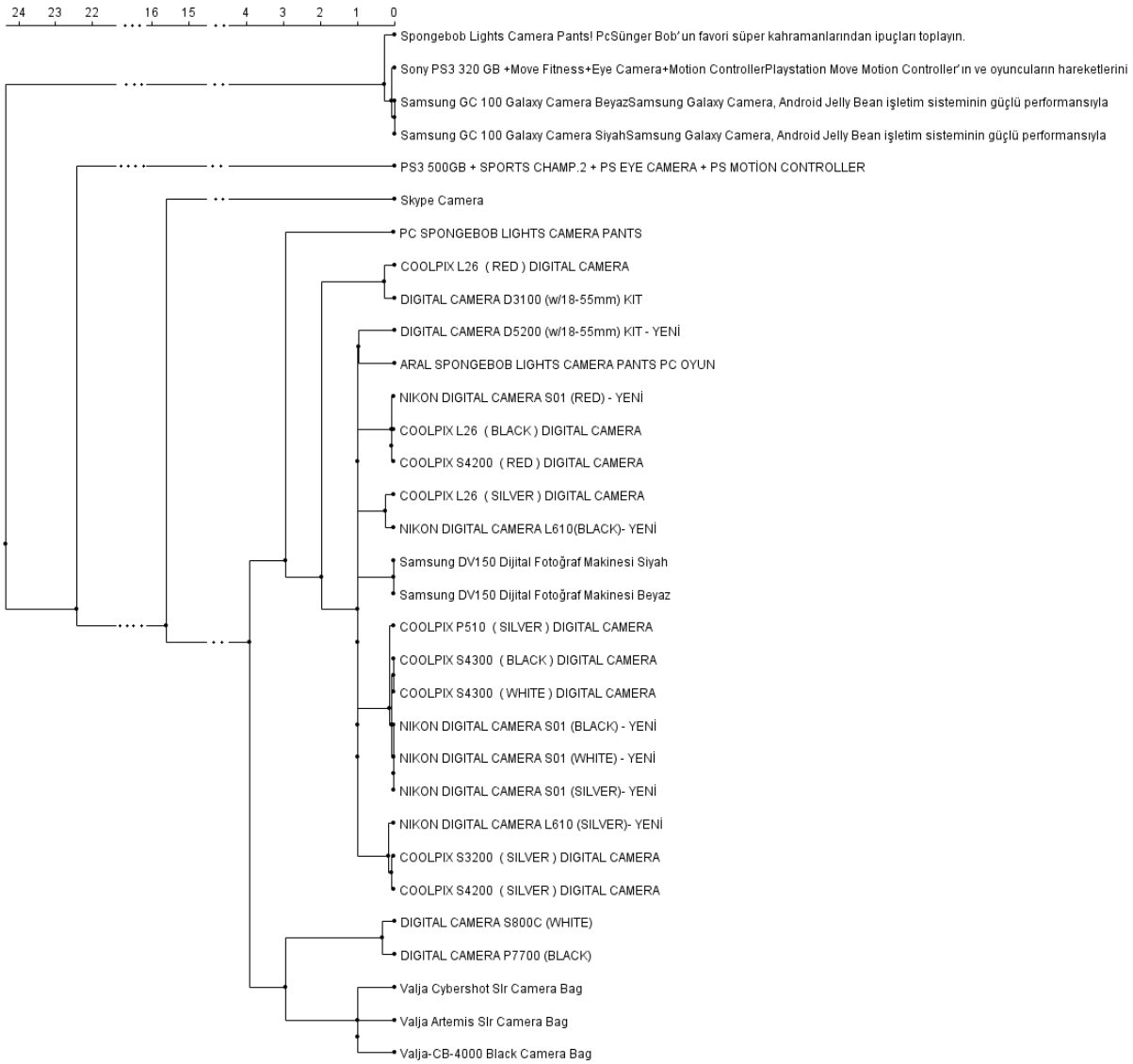

Figure 5. Dendrogram view of "camera" search results.

resources and extracting it take a long time. Before the improvement, it was taking 5 minutes to gather the data. After the improvement has been performed, the gathering time decreased as low as 2 minutes. This is a very important step to optimize the efficiency for SAGO android shopping application. The time which is 2 minutes is only the collection time from internet resources, and extra time is also needed for extracting the data. Moreover, some data is getting lost due to the high response time of the internet resources. Many of these resources do not have a web service therefore the data does not provide data limited time.

Jsoup which is a Java HTML Parser for collecting data from internet resources is used in the SAGO mobile shopping Android application. This parser connects to the internet resources via HTTP protocol and collects the data. Each connection to the internet resources has a timeout period. Timeout is 3 seconds by default. When the SAGO application collects the data by Jsoup, the response time of the internet resources changes and it exceeds the timeout thus the connection goes down and data could not be retrieved. This problem is handled by increasing the timeout to 5 seconds. Changing the timeout showed us that if the timeout was increased a higher value, for example 10 seconds, Jsoup does not lose any data. However, this change affects the total collection time. It takes approximately 2 minutes to gather data from five internet resources by using the default timeout. If the timeout is increased, the total collection time exceeds 2 minutes. After some observations have been made, 5

seconds have been set as a timeout, and the total collection time will be 5 minutes.

For the demonstration purposes, when the location of mobile device is in Ataşehir, Istanbul, Turkey, 1210 results will be collected just for searching "iphone". The smart list algorithm is not rendered on this data. There are only three keywords used; they are "iphone", "laptop", and "camera". **Table 1** shows the number of results with keywords from five local electronic stores. In this table, the threshold is 3 seconds. After 3 seconds passes, SAGO stops collecting data from the internet resources.

The total number of data that was collected at the same time from these internet resources is: 44 for searching "camera" keyword in 14 seconds, 800 for searching "iphone" keyword in 2 minutes and 47 seconds, 362 for searching "laptop" keyword in 1 minute and 38 seconds.

Table 2 shows the error ratio of the search result. These search results are provided in **Table 1**. As we can see, the data was collected successfully in the "camera" search keyword. The worst error is in the "iphone" searches in all local brands because the size of data is very big therefore, some data is dismissed. The response time is changed according to both server and search keyword.

For the next demonstration (**Table 3**), when the threshold is removed, more data are gathered; however, the total time is increased. According to experiment results, Electro World responses are quicker than other electronic local stores.

The total number of data is 44 for searching "camera" keyword in 12 seconds, 1211 for searching "iphone" keyword in 3 minutes and 34 seconds, 370 for searching "laptop" keyword in 1 minute and 32 seconds.

After rendering Greedy Search Filtering on the result data, the total number of results is: 34 for searching "camera" keyword, 996 for searching "iphone" keyword, and 146 for searching "laptop" keyword. If the threshold is 3 sec, the total number of results will be: 34 for searching "camera" keyword, 715 for searching "iphone" keyword, and 124 for searching "laptop" keyword. In addition, SAGO gave out memory error in one experiment.

As you can see in **Table 4**, the error rate is close to zero by removing the threshold. This is because of the response time of each server of local electronic stores. This table provides us that these servers do not send the result in expected time.

Figure 6 shows the sample search results for the "laptop" keyword in SAGO.

Table 1. Seaarch result table (Threshold 3s).

Brand Name	# of data for iphone		# of data for laptop		# of data for camera	
	Gathered	Available Data	Gathered	Available Data	Gathered	Available Data
Darty	236	276	80	100	28	28
Vatan	74	76	8	8	4	4
Bimeks	120	298	100	214	4	4
Teknosa	30	362	48	48	1	1
Electro World	200	200	1	1	7	7

Table 2. Error rate of search result (Threshold 3s).

Brand Name	Error Rate (%)		
	iphone	laptop	camera
Darty	14.49	20	0
Vatan	2.63	0	0
Bimeks	59.73	53.27	0
Teknosa	91.71	0	0
Electro World	0	0	0

Table 3. Seaarch result table (Threshold 0s).

Brand Name	# of data for iphone		# of data for laptop		# of data for camera	
	Gathered	Available Data	Gathered	Available Data	Gathered	Available Data
Darty	276	276	100	100	28	28
Vatan	74	76	8	8	4	4
Bimeks	298	298	214	214	4	4
Teknosa	362	362	48	48	1	1
Electro World	200	200	1	1	7	7

Table 4. Error rate of search result (Threshold 0s).

Brand Name	Error Rate (%)		
	iphone	laptop	camera
Darty	0	0	0
Vatan	2,63	0	0
Bimeks	0	0	0
Teknosa	0	0	0
Electro World	0	0	0

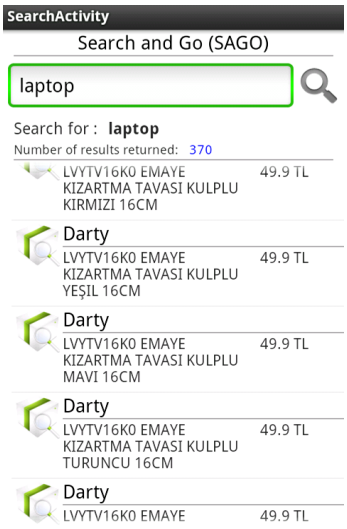

Figure 6. Sample search result.

Consequently, the mobile shopping android application performance results have shown for five local electronic stores. The performance is satisfactory to reach the goals of the designed shopping application. For wider usability, SAGO is a PoC application that can be expanded easily to add other sectors such as local pet shops, book shops, etc. The SAGO is very useful mobile application for mobile users for decision making on purchasing, which can be very beneficial for consumers.

Acknowledgements

This research project would not have been possible without the support of many people. First of all, I wish to express my thankfulness to my adviser, Associate Professor Dr. Oğuz Bayat offered me invaluable assistance,

guidance, and support.

Thanks also to my parents, my friends for their moral support and my team members for providing guidance and advice.

References

[1] Pew Research Center (2011) How People Learn About Their Local Community.

[2] PayPal (2013) Mobilising Sales Making Money in the Mobile Commerce Revolution.

[3] Marker, F. and Chan, Y.H. (2009) A Survey on Android vs. Linux.

[4] Bell, M. (2008) Introduction to Service-Oriented Modeling. In: *Service-Oriented Modeling*: *Service Analysis*, *Design*, *and Architecture*, Wiley & Sons, Hoboken.

[5] Jonathan Hedley (2009-2013) Jsoup Java HTML Parser, with Best of DOM, CSS, and Jquery. http://jsoup.org/

[6] Hastie, T., Tibshirani, R. and Friedman, J. (2009) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Berlin. http://dx.doi.org/10.1007/978-0-387-84858-7

[7] Huang, A. (2008) Similarity Measures for Text Document Clustering. In: *New Zealand Computer Science Research Student Conference* (*NZCSRSC*), Christchurch, April 2008.

[8] Peng, Q., Meng, W., He, H. and Yu, C. (2004) WISE-Cluster: Clustering E-Commerce Search Engines Automatically. WIDM Workshop.

[9] Levenshtein Distance. http://en.wikipedia.org/wiki/Levenshtein_distance

[10] Hamming Distance. http://en.wikipedia.org/wiki/Hamming_distance

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.