San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

1-1-2008

Fuzzy Range Query in XML

Dexiong Zhang
San Jose State University

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_projects

Recommended Citation
Zhang, Dexiong, "Fuzzy Range Query in XML" (2008). *Master's Projects.* Paper 119.

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Fuzzy Range Query in XML

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirement for the Degree

Master of Science

by

Dexiong Terry Zhang

May 2008

© 2008

Dexiong Terry Zhang

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Tseng, Department of Computer Science, San Jose State University

Dr. Robert Chun, Department of Computer Science, San Jose State University

Dr. Mark Stamp, Department of Computer Science, San Jose State University

APPROVED FOR THE UNIVERSITY

ABSTRACT

Fuzzy Range Query in XML

by Dexiong Terry Zhang

This writing project presents a new approach to implement a fuzzy range query solution for retrieving Extensible Markup Language (XML) data. Ever since XML was introduced, it has become a web standard to describe data on the Internet. The need for performing range query against XML data is growing day by day. Many search service providers are eager to improve their solutions on range query against XML data. The project studies and analyzes the limitations on the current range query solutions. The project also proposes a new solution using fuzzy semantic analysis to quantify XML data so that it can be represented within a range. This is accomplished by applying fuzzy logic algorithm to classify and aggregate XML data based on the semantic closeness. An intuitive web interface is also introduced to aid the user to input fuzzy search criteria. Instead of specifying crisp values in the current solutions, the user can simply drag and drop to indicate fuzzy values. Therefore, it’s more user-friendly and desirable for fuzzy range query.

ACKNOLEDGEMENT

I would like to express my appreciation to Dr. Chris Tseng for his technical guidance, invaluable insights, motivation, and resources, without which this project could never have been finished. I would also like to express my gratitude to my committee members Drs. Robert Chun and Mark Stamp for their useful suggestions and comments on my work. Finally, I would like to thank my family and friends for their encouragement and support throughout the entire project process.

Table of Content

1 Introduction 8
2 Project Overview 9
2.1 Evaluation of Current Solutions 9
2.1.1 Restriction on Range Input 9
2.1.2 Ignore User Preference 10
2.1.3 Difficulty on Handling Linguistic Data 10
2.2 Project Goals 11
2.2.1 Fuzzify Range Input 12
2.2.2 Allow User Specified Preference 12
2.2.3 Enhance Linguistic Range Query 13
3 Project Design 14
3.1 Range Query Interface 14
3.2 Fuzzy Search Engine 16
3.2.1 One Level Fuzzification 17
3.2.2 Two Levels Fuzzification 20
3.3 Native XML Database 26
4 Implementation 28
4.1 Technologies and Tools 28
4.1.1 Fuzzy Control Language 28
4.1.2 jFuzzyLogic 30
4.1.3 XAMPP 31
4.1.4 Apache Xindice 31
4.1.5 Apache Tomcat 33
4.1.6 PHP/Java Bridge 33
4.1.7 script.aculo.us 35
4.2 Experimental Result 35
4.2.1 Test Scenario 36
4.2.2 Fuzzy Numerical Range Query 37
4.2.3 Fuzzy Linguistic Range Query 39
4.2.4 Combination of Fuzzy Numerical and Linguistic Range Query 42
5 Conclusion and Future Enhancement 45
5.1 Conclusion 45
5.2 Future Enhancement 45
6 Reference 47
7 Appendix: Search Data Pool 49

List of Figures

Figure 1: A typical range query interface on price 10
Figure 2: A typical interface for linguistic search 11
Figure 3: Range query solution architecture 14
Figure 4: Sample linguistic range query interface 15
Figure 5: Double slider interface example 15
Figure 6: Membership function for search range 18
Figure 7: Membership function for preference value 18
Figure 8: Membership function for score 19
Figure 9: Fuzzy rule for one level fuzzification 19
Figure 10: Defuzzification of score 20
Figure 11: Fuzzification of price 21
Figure 12: Fuzzification of `inputRange` membership degree 22
Figure 13: Fuzzification of `preference` membership degree 23
Figure 14: Membership functions for `score` defuzzification 24
Figure 15: Two level fuzzification rule set 25
Figure 16: Fuzzy rule set matrix 25
Figure 17: Score value defuzzified by region 26
Figure 18: Fuzzy Control Language structure 29
Figure 19: Collections in Apache Xindice 32
Figure 20: XML document in collection 32
Figure 21: Usage example of PHP/Java Bridge 34
Figure 22: Testing data in Apache Xindice 36
Figure 23: Fuzzy numerical range query interface 37
Figure 24: Fuzzy numerical range query result 38
Figure 25: Fuzzy linguistic range query interface 40
Figure 26: Fuzzy linguistic range query result 41
Figure 27: Fuzzy numerical and linguistic range query interface 42
Figure 28: Fuzzy numerical and linguistic range query result 44

1 Introduction

We are living in the century of information. Human has never relied so much on information before. Everyday, millions of people will surf on the web and look for all kinds of information. Many websites survive by satisfying the need of these people. These websites provide some kind of service to help people to search the information they want. The key of success for these websites is the process of query should appear to be simple to the users and the result of the query should be accurate. This means an easy to use interface and a powerful search engine are required.

As the Internet evolves, information stored in Extensible Markup Language (XML) structure becomes popular on the web. Current solutions for search engine are facing some difficulties on query, especially range query, against loosely structured XML data. This project seeks to provide a better solution on both interface and search engine to handle range query. By applying fuzzy logic to search engine and introducing an intuitive interface, the project presents a new solution for range query against XML data. The solution intends to address the need of range query from the end-users' perspective and is designed to be adopted in various domain. An implementation of the solution is provided to demonstrate the query result is desirable and meaningful to the end-users.

The rest of the report is organized as following:

- Section 2: Discusses limitations on current solutions and proposes project goal.
- Section 3: Explains the project design including high level architecture and detail design of each component.
- Section 4: Illustrates various technologies and tools used in the implementation and presents the experimental result.
- Section 5: Outlines the contributions and suggests future work for the project.

2 Project Overview

In order to come up with a better solution on range query, we first need to study and analyze the limitation of the solutions currently available. The aspects which need to be improved in the current solutions are identified. Then techniques can be introduced to achieve the improvement on these aspects.

2.1 Evaluation of Current Solutions

Currently, there are many search engines available on the Internet. However, not much of them can handle range query very well. One of the major limitations is the restriction on the range input. Another limitation is the ignorance on the user preference. Last but not least is the difficulty on handling linguistic data.

2.1.1 Restriction on Range Input

A common way to specify search criteria in search engine with range query is accepting numerical input from the users. One of the characteristic of this approach is it can only allow crisp input. This means the user has to input or choose a specified value for the query and the result generated will only match the crisp input the user specified. A very good example will be the product search based on price. This kind of service is available on most of the website. Figure 1 shows a common interface of range query service to search for products based on their price. The user is asked to type in the value of the range of his/her expected price. If the user specified the range to be from $200 to $500, the generated result will include all products with a price between $200 and $500 inclusively. This makes sense from a technical point of view. In reality, products are more likely to be listed with a price in the form of $199.99, $199.95 or even $195. These products will not be included in the search result when using traditional search engine with the input sample mentioned above. However, from the end-users' perspective, many people are willing to buy a product with a price of $199.99 and even $195 although the

specified lower bound is $200 when they are shopping for products. Also, many people may consider $505 is the same as $500 although the product price is greater than the specified upper bound. The direct impact to the end-user is he/she may not find the desired product because this limitation on the current search engine.

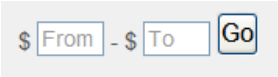


Figure 1: A typical range query interface on price

2.1.2 Ignore User Preference

For most search engine currently available, the search result of a range query is supposed to include a list of items that with search criteria fall into the specified range. This search result is usually sorted and presented in an increasing or decreasing order based on the search criteria. This can be easily done from technical point of view but may not be desirable from the end-users' perspective. Sometime people may have a preference in mind when searching for products. For example, a user may want to search for products with price between $200 and $500. The user may have a concern that products with price lower than $200 have poor quality and products with price higher than $500 are way too expensive for his/her budge. In this situation, the user may consider products with price of $300 are more desirable because the products have reasonably good quality and he/she doesn't need to spend too much money. However, when using search engine currently available, the search result will be sorted either from $200 to $500 or from $500 to $200. This means the user preferred products will not be returned first and the user has to scan through the result to look for the preferred products. This redundant work may bring frustration to the user when he/she has to search through the result.

2.1.3 Difficulty on Handling Linguistic Data

One very obvious characteristic for most of the currently using search engines is they have difficulty on handling linguistic data. To query against linguistic data, most of the

search engines will use exact match instead of range match. For example, Figure 2 shows an interface for searching wine based on its taste. The user can specify the taste of desired wine by selecting the check box in front of the wine taste. The search result will be a list of wine with taste matching the specified input. Although this approach can be used for range query against linguistic data, it is not very intuitive to specify a range and may cause some confusion. In this example, the user has to select all wine taste within the range he/she wants. If the user wants the search for wine with taste between “Extra Dry” and “Semi Sweet”, he/she has to select all check boxes for “Extra Dry”, “Semi Dry” and “Semi Sweet”. If he/she selects only “Extra Dry” and “Semi Sweet”, the search engine cannot produce the result he/she expected. A more serious problem is that this kind of search engine cannot address the fuzziness of linguistic terms. The current solution requires all the linguistic terms to be specified in the interface. If a wine has a taste between “Extra Dry” and “Semi Dry”, it has to be classified using other linguistic term like “Somehow Dry” and be specified in the interface. Otherwise, this wine cannot be found by any query. The impact is the interface will be awkward if the available selection is huge. For instance, if there are more than twenty different wine tastes available, this interface will create confusion for the user.

Figure 2: A typical interface for linguistic search

2.2 Project Goals

The objective of this project is to provide an alternative solution to the current search engine with respect to range query. It is important to understand that this project is not

trying to develop a solution to replace any existing solutions on range query under any circumstance. Rather this project seeks for a solution to address the limitation of the current search engine on range query hence provide an enhanced alternative. The project is developed with the following goals.

2.2.1 Fuzzify Range Input

Since the current search engines enforce restriction on the range input, only items with search criteria fall into the specified search range will be returned. The proposed solution addresses this issue by including reasonable amount of items, which with search criteria fall outside but still close to the user specified search range, to be returned in the search result. The key is allowing fuzzy input from the user and using fuzzy logic to fuzzify the range input. By doing so, the user input is no longer crisp and the amount of return result that satisfied the range input can be adjustable.

2.2.2 Allow User Specified Preference

The proposed solution also addresses the issue of the inability to specify user preference in current range query solutions. Current range query solution can only return search result in ascending or descending order. Including a preference value can provide additional control based on the user interest [1]. By allowing the user to specify a preference value on the range input, the proposed solution returns the search result in order of “closeness” to the preference value. This “closeness” value can be considered as the ranking of a product and it is achieved by using fuzzy logic to determine the relationship between the preference value and the actual data value [2]. Combining the fuzzy algorithm for the range query and the preference value, the ranking value can represent the desirability of each search item. The search result then returns the items in descending order by the ranking value therefore the query result is more desirable to the user.

2.2.3 Enhance Linguistic Range Query

The range query solution from many search engines currently available has limitation on querying linguistic data. Both search engine and interface cannot address the need for linguistic range query very well. A linguistic range query solution is converted from the proposed fuzzy numerical range query solution. This conversion enables the proposed solution to handle linguistic data. Enhancement is done to both search engine and interface to increase the usability. The user is now able to fuzzily specify the linguistic range with preference value to obtain desirable search result.

3 Project Design

For this project, a fuzzy range query solution is introduced and developed. The solution contains three major components which include a range query interface, a fuzzy search engine, and a native XML database. The range query interface collects input from the user and sends it to the fuzzy search engine. The search engine generates the query and sends the query to the native XML database to retrieve data. The data is sent back to the search engine. The search engine then fine tune the data to create the search result and sent it to the interface. Then the interface will output the result to the user. Figure 3 shows the high level architecture of the solution.

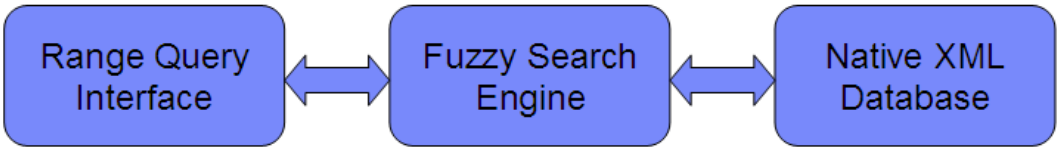


Figure 3: Range query solution architecture

3.1 Range Query Interface

The range query interface is designed to be intuitive and user-friendly. Since the interface is supposed to be used on both numerical range query and linguistic range query, it needs to be able to collect input as both numerical and linguistic terms. Besides this, the interface needs to be simple enough since reducing the structure complexity of the interface can improve the efficiency of the interface [3]. A simple approach for designing the interface is to allow the user to type in the range value such as in Figure 1. By using input fields, the user can key in both numerical and linguistic values. However, this creates ambiguity because the user can type in anything he/she wants. This means the input data from the user cannot be guaranteed to be valid. Hence simply using input field for the interface is not a desirable approach.

Another common approach for the range query interface is to let the user selecting from drop down menus. An example is shown in Figure 4. The user can use this interface to search on items with size between medium and big. Notice the user can also specify "Between Big and Medium" instead of "Between Medium and Big". It does not necessary mean that the user has preference on the first linguistic term over the second one. The major disadvantage of this interface is that it can only take crisp input. For linguistic range query, all available linguistic terms have to be added to the drop down menu so that they can be selected by the user. For numerical range query, the drop down menu can only allow the user to specify limited number of predefined ranges. This highly reduces the flexibility of the interface so that drop down menu cannot satisfy the need for the range query interface.

Figure 4: Sample linguistic range query interface

The ideal approach for the range query interface is to use a slider interface which can accept range input. A traditional slider interface includes one slide track and one slide bar which prohibits the interface from accepting range input. But using double slider with one slide track and two slide bars can allow the user to specify a range. Such interface is illustrated in Figure 5. In this example, the user can specify the price lower bound of the input range using the left slide bar and specify the price upper bound of the input range using the right slide bar. It also shows the price representation of each slide bars regarding to their position on the slide track. Notice this interface can only accept crisp input since the slide bars move at a $25 increment. This is a limitation which can be easily removed.

Figure 5: Double slider interface example

The final design of the fuzzy range query interface for this project is inspired from the double slider interface above. The fuzzy range query interface has one slide track and three slide bars. The length of the slide track indicates the maximum size of the available search range. The slide bar on the left is used to specify the lower bound of the input range and the slide bar on the right is used to specify the upper bound of the input range. The user can use the slide bar in the middle to specify the preference value. None of the slide bar can be moved cross other slide bars. This restriction can ensure that the preference value is set to be within the specified range hence eliminating ambiguity. This interface can also be used to fuzzily accept linguistic input. To do so, the lowest available linguistic value and the highest available linguistic value in the search pool need to be identified and presented on the interface. This way, the length of the slide track represents entire linguistic range that is available for searching. Then the user can use the two slide bars on the outside to specify a fuzzy range on the linguistic data. The slide bar in the middle can be used to fuzzily specify a preferred linguistic value. Therefore the interface can be used to handle both numerical and linguistic range query. This approach provides a fuzzy range query interface which is more flexible and intuitive than tradition interface using input fields or drop down menus.

3.2 Fuzzy Search Engine

Fuzzy logic has been used in various applications for a long time. In the project, fuzzy algorithm is embedded in the search engine to generate more desirable search result. Since the search engine is designed to perform range query, the choice of fuzzy algorithm is also intended to optimize the performance of range query. Two different designs are included in the solution. Both designs can generate reasonable result but their performance is slightly different. The choice of implementation on the designs should be based on the application and search domain.

3.2.1 One Level Fuzzification

The first design is similar to other common approach in fuzzy logic implementation. This design uses one level fuzzification and it defines two membership functions base on the user specify range and preference value. Then the membership degrees of the two functions are aggregated. Another membership function is used to define a score for the search item. The score value is used to rank the search items which means items with higher score will show up before items with lower score. The score is defuzzified based on fuzzy rule set and is used later to sort the search result. To illustrate the design, an example of range query based on price is used. In the example, the search engine performs range query on a data pool with minimum price is $0 and maximum price is $100. The user specifies the search range to have a lower bound of $40 and an upper bound of $70. The user preferred value is set to be $60.

The membership function of the user specified search range over price is defined with a trapezoid function. As shown in Figure 6, the `inRange` function calculate the membership degree of an item based on whether the item has a price fall within the user specified range. For items with price between $40 and $70, they will have an `inRange` membership degree of 1 which means the price of these items are definitely within the user specified range. The `inRange` membership degree will decrease rapidly when the price of an item is getting greater than the price upper bound or smaller than the price lower bound. This indicates that the price of the item is moving away from the user specified range. Ultimately, items with price which is too far away from the search range will receive a membership degree of 0. For these items, their price will certainly be considered as outside of the user specified range.

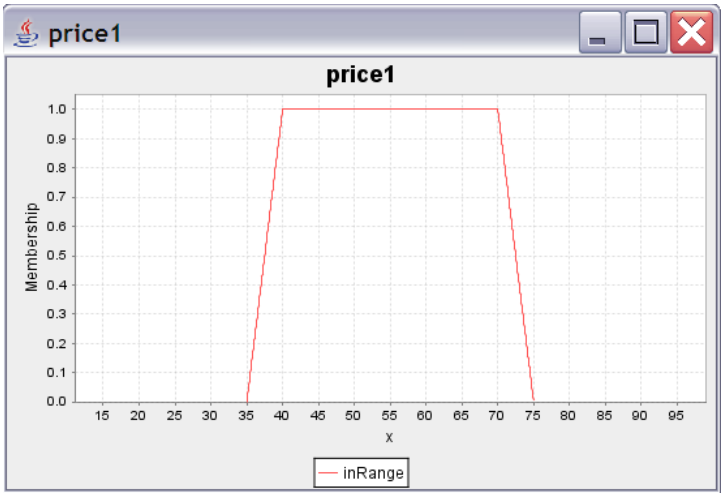


Figure 6: Membership function for search range

Another membership function is used to define the preferred price specified by the user. This `preference` membership function is a Gaussian bell function as shown in Figure 7. It indicates how far an item is away from the preferred price. The function has a peak at $60 which means an item will have a membership degree of 1 if it has a price of $60. Item with price greater or less than $60 will have its membership degree determined based on the distance to the preference value. Notice that the price variable in this function uses a different name than the price variable in the `inRange` membership function. This is necessary and will be explained later in this section.

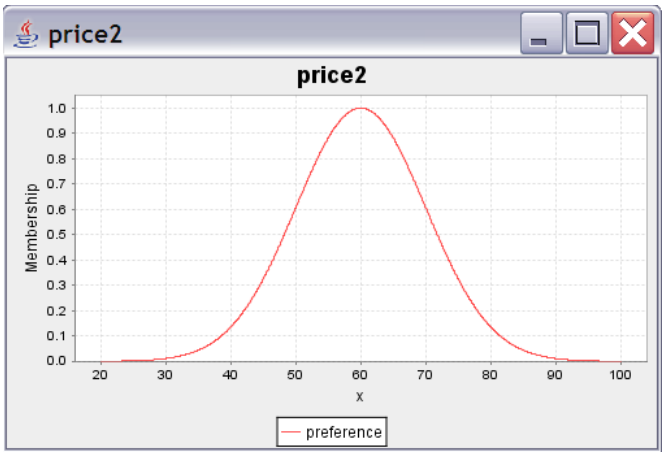


Figure 7: Membership function for preference value

One more membership function is used to define the ranking of an item. The `result` function in Figure 8 is used to defuzzify the `score` of an item from the `inRange` membership degree and the `preference` membership degree of the item. The `inRange` membership degree and the `preference` membership degree are aggregated together based on the fuzzy rule set explained later. The `result` membership degree will then be defuzzified in to numerical `score` value. In this example, the minimum `score` is 0 and the maximum `score` is 100.

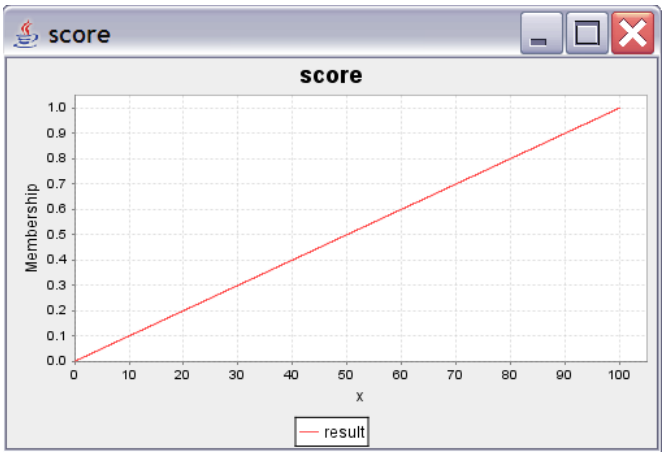


Figure 8: Membership function for score

In this design, only one simple rule, which is shown in Figure 9, is needed. The rule is designed to be fired every time so that a `score` value will always be returned. The rule use an `AND` operation which means when both `price1 IS inRange` and `price2 IS preference` conditions are satisfied, a `score` value will be generated. As mentioned above, the `price1` and `price2` variables actually have the same value. If only one "price" variable is used, we will have two membership functions for this "price" variable. Then we will not be able to use the `AND` operation since `AND` operation doesn't allow two conditions from the same variable.

```
IF price1 IS inRange AND price2 IS preference THEN score IS
result
```

Figure 9: Fuzzy rule for one level fuzzification

After fuzzy rule is fired, the `return` membership function will be defuzzified with a shape as shown in Figure 10. In this example, the shape is for an item with a price of $50 and it obtains a `score` of 60.7. The defuzzification process uses Left Most Max method which determines the defuzzification value by the first maximum membership degree from the left side of the shape [4]. In this example, the maximum membership degree is 0.61 and the first value from left side to have a membership degree of 0.61 is 60.7. Therefore, the final score is 60.7.

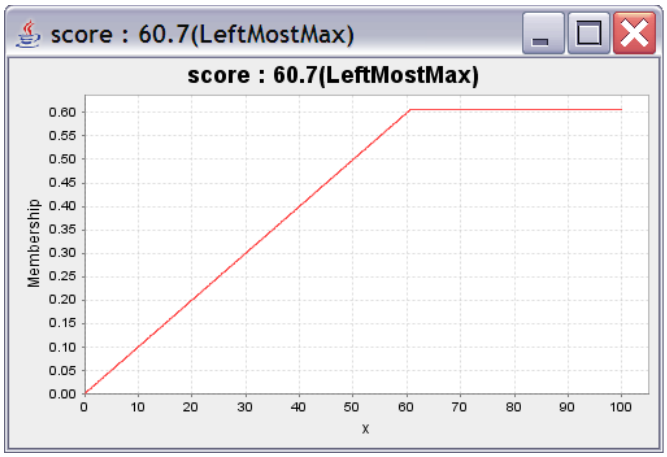


Figure 10: Defuzzification of score

This design of the fuzzy search engine has a behavior of always firing rule. Also its population density of the score tends to be higher on the two ends of the search range. This means the score value for items with very high score are close to each other and so as for the score value for items with very low score. Therefore, this design performs better when the search pool has lower density.

3.2.2 Two Levels Fuzzification

The other design is revised from the one level fuzzification. This design uses two levels fuzzification. In this approach, the data value first will be fuzzified to obtain the membership degree. Then the membership degree will be fuzzified again based on other

membership functions. The idea is to classify data value into different categories. Let's use the same example as the one level fuzzification design to show the difference of the two designs. We have a data pool with minimum price is $0 and maximum price is $100. The lower bound, preference value and upper bound are set to be $40, $60 and $70 respectively.

In this design, we have only one price variable which includes two membership functions as shown in Figure 11. The `inputRange` membership function is defined as a trapezoid function and the `preference` membership function is defined as Gaussian bell function. The `inputRange` function determines whether the price of an item is within the user specified range. When an item has a price which is between $40 and $70, the item gets a membership degree of 1. For items with price fall outside of the user specified range, their membership degree is calculated base on the shape of the trapezoid. The `preference` membership degree is calculated to reflect the distance between the price of an item and the preferred price. When an item has a price of $60, its `preference` membership degree is 1. When the price of an item is getting further away from the preferred price, its `preference` membership degree will get smaller.

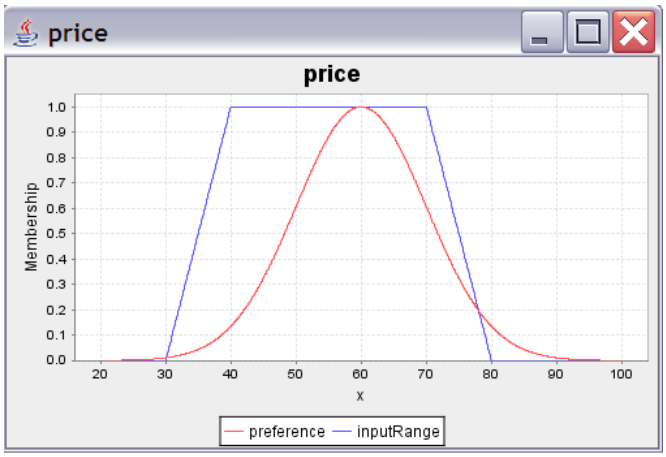


Figure 11: Fuzzification of price

After we obtain both the `inputRange` and the `preference` membership degrees, we can use these two membership degree values as input for further fuzzification. For the

`inputRange` membership degree, we classify it into three different categories with three membership functions:

- `inside`: Represents items with price fall within the user specified range.
- `onedge`: Represents items with price fall outside of but still close to the user specified range.
- `outside`: Represents items with price which is far away from the user specified range.

As we can see in Figure 12, an item with an `inputRange` membership degree at 0.9 will receive a 0.6 membership degree on `inside`, a 0.4 membership degree on `onedge`, and a 0 membership degree on `outside`. This means the item is more likely to be considered as `inside` over `onedge` and it will not be considered as `outside`. Therefore, the price of the item is more likely to be within the user specified price range. On the other hand, if an item has an `inputRange` membership degree at 0.6, its `inside` membership degree will be 0 which means the item will not be considered as `inside`. The `outside` and `onedge` membership degree will be 0.6 and 0.4 which means the item is more likely to be `outside` than `onedge`. The price of the item then can be considered far away from the user specified price range.

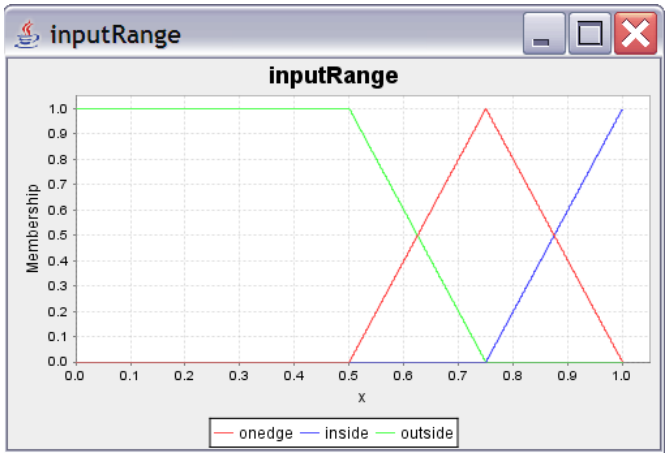


Figure 12: Fuzzification of `inputRange` membership degree

The `preference` membership degree is also fuzzified again. Three different membership functions are used for the fuzzification.

- `close`: The price of the item is close to the user specified preferred price.
- `middle`: The price of the item is not so close but also not too far from the user specified preferred price.
- `far`: The price of the item is far away from the user specified preferred price.

In Figure 13, if an item has a `preference` membership degree of 0.9, its `far` membership degree will be 0 which means the price of the item is definitely not too far from the user specified preferred price. The item's `close` membership degree will be 0.8 and its `middle` membership degree will be 0.2. This means the item will be considered more as `close` than `middle` and the price of the item should be really close to the user specified preferred price. In contrast, if the `preference` membership degree of an item is 0.3, it will get a 0 on the `close` membership degree which indicates the price of the item is certainly not close to the user specified preferred price. However, since the item will get a 0.6 on the `middle` membership degree and a 0.4 on the `far` membership degree, the item is more likely to be considered as `middle` than `far`. This means the price of the item is slightly far away from the user specified preferred price.

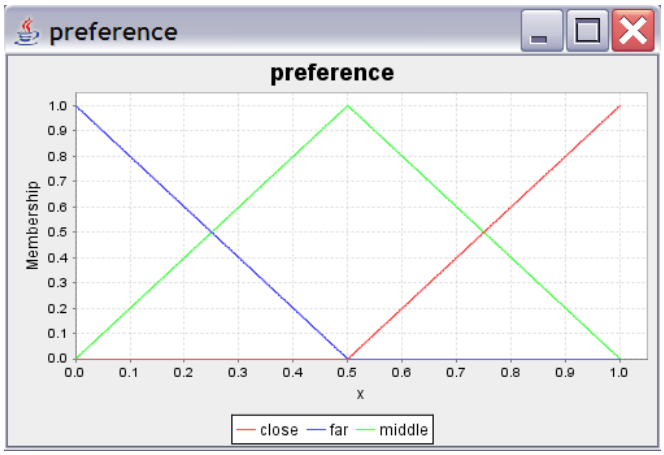


Figure 13: Fuzzification of `preference` membership degree

Once all the membership functions for fuzzification are defined, we need to define membership functions to defuzzify the `score` of an item. The `score` can have a value from 0 to 100 and there are five membership functions for the `score` value as shown in Figure 14.

- `Veryhigh`: Provides more weight for the very high portion in the `score` range.
- `High`: Provides more weight for the high portion in the `score` range.
- `Middle`: Provides more weight for the middle portion in the `score` range.
- `Low`: Provides more weight for the low portion in the `score` range.
- `Verylow`: Provides more weight for the very low portion in the `score` range.

The membership degree of these five membership functions will be set according to the fuzzy rule set. Then the final value of the `score` is defuzzified by the shape created by these five membership functions.

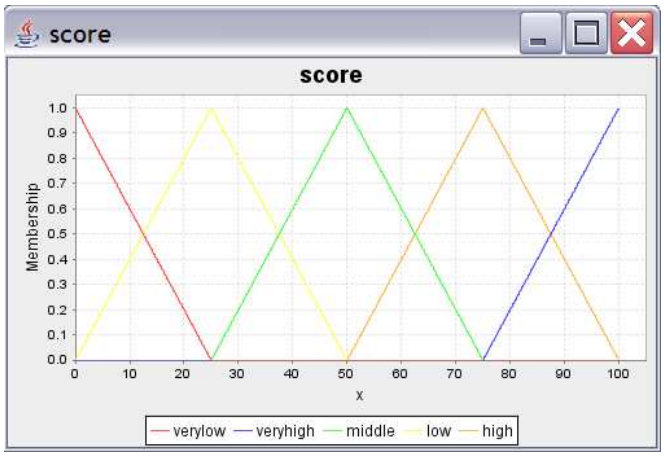


Figure 14: Membership functions for `score` defuzzification

The defuzzification process is made by the rule set. In this design, the rules are `AND` operations between the `inputRange` membership degree and the `preference` membership degree. For example, if an item's `inputRange` membership degree is fuzzified as `inside` and its `preference` membership degree is fuzzified as `close`, the item will get a `veryhigh` on its `score`. After all rules are tested, the membership

functions of the `score` can be used to defuzzify the final score value. The detail rule set is illustrated in Figure 15 and 16.

```
IF inputRange IS inside AND preference IS close THEN score IS veryhigh;
IF inputRange IS inside AND preference IS middle THEN score IS high;
IF inputRange IS inside AND preference IS far THEN score IS middle;
IF inputRange IS onedge AND preference IS close THEN score IS middle;
IF inputRange IS onedge AND preference IS middle THEN score IS low;
IF inputRange IS onedge AND preference IS far THEN score IS verylow;
```

Figure 15: Two level fuzzification rule set

		inputRange	
		onedge	inside
preference	far	verylow	middle
	middle	low	high
	close	middle	veryhigh

Figure 16: Fuzzy rule set matrix

The final score is calculated by the five membership functions of `score`. These five membership functions form a shape based on their membership degree. Figure 17 shows the defuzzification graph created when the item has a price of $50. After the shape is created, the region of the shape is defuzzified into one single score value by using the Center Of Gravity method. The Center Of Gravity method first calculates the center of the shape and then finds its representation on the horizontal axis [4]. This value on the horizontal axis is then used as the final score value. For the shape in Figure 17, it creates a final score of 75.52.

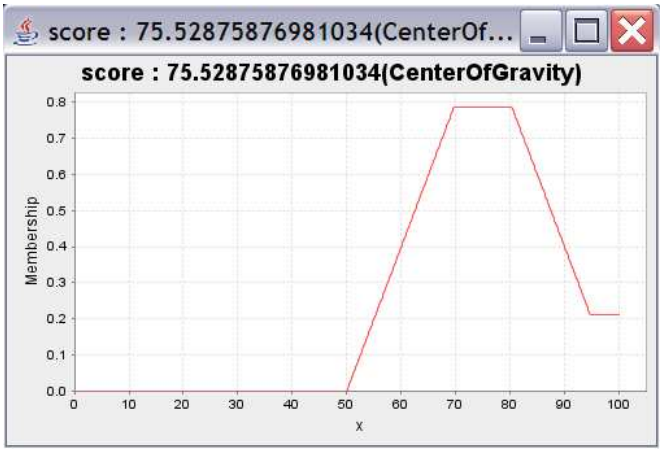


Figure 17: Score value defuzzified by region

The two levels fuzzification design allows the search items to be classified into different categories. The first level fuzzification turns the input data value into membership degree. The second level fuzzification then fuzzify the membership degree come from the first level fuzzification. This design can generate the score value tend to be evenly distributed over the search range therefore the design can perform well for most of the applications.

3.3 Native XML Database

Since this project is intended to query against XML data, a media to store XML data is needed. The traditional way is that we can simply leave the XML data in XML document and then load the XML document to parse the XML data when the data is needed. A new approach is storing the XML data in native XML database. XML data is stored in its hierarchical form in native XML database. This means industry standard technologies such as XPath and XQuery can be used directly against the data in the database. Also, indexing is available in native XML database which provide fast data access. Native XML database also has advantage over relational database. Before native XML database was introduced, people used to shred XML data into relational database. This approach loses the flexibility given by the XML structure. Native XML database overcome this disadvantage by storing XML data in its original hierarchical structure. Since XML data

can be easily manipulated with native XML database, the complexity of the application on XML parsing is also reduced [5, 6, 7]. With all these advantages, a native XML database is used as the storage for testing XML data set in this project. The native XML database chosen for the project is an open source application and will be covered in the later section.

4 Implementation

The implementation of this project is a web application which allows the user to perform range query against XML data. It simulates the common search engine web service environment. The interface is implemented with HTML and JavaScript. The search engine is built on PHP and Java. The database is an existing database product which supports native XML storage. Various technologies and tools are used in the implementation process and they will be explained clearly in this section. Also, the experimental result is given and illustrated.

4.1 Technologies and Tools

A wide range of technologies and tools are used to build the project application. Some of them are commonly used and the others are relatively new for web applications. These technologies and tools are covered in details to provide a better understanding on the implementation process.

4.1.1 Fuzzy Control Language

One of the most important components of the application is the fuzzy search engine. In this implementation, the fuzzy algorithm in the search engine is built with Fuzzy Control Language (FCL). The Fuzzy Control Language was introduced by the International Electrotechical Commission Technical Committee and was designed to provide a way to control over system that without an explicit process model [8]. With Fuzzy Control Language, fuzzy logic application can be directly implemented in a way which is more human understandable. The Fuzzy Control Language defines fuzzy logic in function block. Each function block contains a complete fuzzy inference system. The fuzzy inference system can include input variables, output variables, fuzzification of the input variables, defuzzification of the output variables, and the fuzzy rule set. Figure 18 shows the structure of Fuzzy Control Language.

```
FUNCTION_BLOCK <name>

VAR_INPUT
     <variable name> REAL;
END_VAR

VAR_OUTPUT
     <variable name> REAL;
END_VAR

FUZZIFY <variable name>
     TERM <term name> := <term definition>;
END_FUZZIFY

DEFUZZIFY <variable name>
     TERM <term name> := <term definition>;
     METHOD: <defuzzification method>;
END_DEFUZZIFY

RULEBLOCK
     <operator> : <algorithm>;
     ACCUM : <accumulation method>;
     RULE : IF <condition> THEN <condition>;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Figure 18: Fuzzy Control Language structure

In Fuzzy Control Language, the individual set of the variables are referred as term and terms are defined with membership functions. The fuzzification and defuzzification can support various membership functions including both continuous and discrete. Various defuzzification methods are supported including Left Most Max, Center of Gravity, and so on. The operator defines how the algorithm to process different conditions in a rule. The accumulation method is used to aggregate different terms together. A rule block can contain one or more rules and multiple function blocks can be defined depend on the needs of the application. The Fuzzy Control Language provides a standard approach to define fuzzy application and can be implemented across various platforms.

4.1.2 jFuzzyLogic

The Fuzzy Control Language cannot work stand alone. It needs to be implemented with other programming language to form a complete package. The jFuzzyLogic is a Java implementation of the Fuzzy Control Language which can be used by Java developers to build fuzzy logic applications [9]. Integrating jFuzzyLogic into other Java applications is very simple. A developer can utilize jFuzzyLogic in a Java program by importing these classes:

```
jFuzzyLogic.FIS
jFuzzyLogic.rule.FuzzyRuleSet
```

Then the sources file which written in Fuzzy Control Language can be loaded into the fuzzy inference system by:

```
FIS.load(filename);
```

Once the sources file is loaded, the function block can be read by:

```
FIS.getFuzzyRuleSet(function_block_name);
```

Now the fuzzy rule set is ready to use. We can pass input to the fuzzy inference system by:

```
FuzzyRuleSet.setVariable(variable_name, value);
```

After all the necessary inputs are set, we can run the fuzzy rule set with the following method:

```
FuzzyRuleSet.evaluate();
```

The output of the fuzzy rule set can be obtained by:

```
FuzzyRuleSet.getVariable(variable_name);
```

Now we have a complete fuzzy logic engine. The jFuzzyLogic package is easy to use and it is open source therefore integrating jFuzzyLogic into this project is an ideal approach to achieve a fuzzy search engine.

4.1.3 XAMPP

The implementation of this project is a web application so a web application server is needed. The web application is developed in PHP so the web application server need to have PHP support. One of such web application server is the XAMPP package from Apache Friends project [10]. The Apache Friends project is intended to promote the Apache web server. The XAMPP package includes a bundle of MySQL, PHP, and Pearl. It supports both PHP 5 and PHP 4 and can be easily switched between these two versions. To be consistent, only PHP 5 was used in this implementation. The implementation is developed on Windows XP therefore XAMPP for Windows is installed to provide web server functionality. XAMPP provides a package for easy setup and maintenance on Windows platform. For this implementation, the XML-DOM and XML-RPC services in PHP 5 are needed and turned on.

4.1.4 Apache Xindice

The database used in this implementation is a native XML database which is called Apache Xindice. Xindice is pronounced zeen-dee-chay and it is a subproject of the open source project Apache [11]. The benefit of using native XML database is we don't need to worry about conversion between XML and other data structure. This is especially useful when we have a very complex XML document which would be extremely hard to map into a traditional database. For easy data access, Xindice provide a XML:DB API for Java development and a XML-RPC API for other languages. In this implementation, the XML-RPC API is used.

Setting up the Apache Xindice is somehow tricky. Apache Xindice is not a standalone application. It requires Java SDK and Apache Tomcat to be installed in the system. It's important to remember to set the environment variable such as `XINDICE_HOME`, `JAVA_HOME` and `CATALINA_HOME`. They are used as place holders to indicate the location of the library files. It is also important to include the jar files in the Xindice

folder to the PATH variable. In Apache Xindice, XML documents are stored in collections. Collection can be considered as table in relational database and each collection can have multiple XML documents. Figure 19 and 20 show the web interface of the Apache Xindice database.

db -

atom
jobs
system
nba
food
addressbook
countries
invoices
elements
products
meta

Figure 19: Collections in Apache Xindice

db - addressbook -

address1
address2

Document 'address1'

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  - Licensed to the Apache Software Foundation (ASF) under one or more
  - contributor license agreements.  See the NOTICE file distributed with
  - this work for additional information regarding copyright ownership.
  - The ASF licenses this file to You under the Apache License, Version 2.0
  - (the "License"); you may not use this file except in compliance with
  - the License.  You may obtain a copy of the License at
  -
  -     http://www.apache.org/licenses/LICENSE-2.0
  -
  - Unless required by applicable law or agreed to in writing, software
  - distributed under the License is distributed on an "AS IS" BASIS,
  - WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  - See the License for the specific language governing permissions and
  - limitations under the License.
  -
  - $Id: address1.xml 511428 2007-02-25 03:30:52Z vgritsenko $
  -->
<person>
  <fname>John</fname>
  <lname>Smith</lname>
  <phone type="work">563-456-7890</phone>
  <phone type="home">534-567-8901</phone>
  <email type="home">jsmith@somemail.com</email>
  <email type="work">john@lovesushi.com</email>
  <address type="home">34 S. Colon St.</address>
  <address type="work">9967 W. Shrimp Ave.</address>
</person>
```

Figure 20: XML document in collection

4.1.5 Apache Tomcat

Since Apache Xindice cannot work stand alone, Apache Tomcat is also installed for this implementation. Just as Apache Xindice, Apache Tomcat is an open source project from Apache Software Foundation [12]. Apache Tomcat is an application server that implements the Java Servlet and provides an environment to run Java code on web server. Apache Tomcat can be installed standalone, but XAMPP also provides an Apache Tomcat extension. Since XAMPP was used in the development environment, using its Apache Tomcat extension should be a good idea. Installing the Apache Tomcat extension in XAMPP is straightforward. An important step after the installation is to make sure the `CATALINA_HOME` variable is set as it is needed for Apache Xindice.

4.1.6 PHP/Java Bridge

In this implementation, the fuzzy search engine is developed using jFuzzyLogic tool kit which uses a Java API. But the connection between the interface and database is written in PHP. Therefore, a way to connect the interface and database to the fuzzy search engine is needed so that these three components can communicate with each other. To achieve this goal, another open source tool kit, which can bridge between the two languages PHP and Java, is used. This tool kit is called PHP/Java Bridge. It allows the user to access Java classes within PHP script. Or the user can access PHP script within Java classes [13]. Therefore, it provides a fast and flexible way to communicate and reuse resource between the two languages. There is a PHP Java extension that was introduced in PHP 4. This extension provides a simple way to access Java objects from PHP. However, this extension is only experimental and it is slow and unstable. Therefore, the PHP Java extension is no longer supported in PHP 5 and using PHP/Java Bridge is a better solution. Figure 21 shows a simple way to access Java resource in PHP using PHP/Java Bridge.

```
<?php

// get instance of Java class java.lang.System in PHP
$system = new Java('java.lang.System');

// demonstrate property access
echo 'Java version=' . $system->getProperty('java.version')
. '<br/>';

?>
```

Figure 21: Usage example of PHP/Java Bridge

There are many different ways to set up the PHP/Java Bridge. However, none of them is very straightforward and installation is quite different based on the operating system, the web server and the application. In this implementation, the PHP/Java Bridge is installed on Windows XP with XAMPP. Following is the steps I used to install this tool kit:

1. First of all, make sure Java VM is correctly installed and the necessary Java environment variables are set.
2. Download and extract the PHP/Java Bridge package into a local folder.
3. Create an empty folder called `JavaBridge` and two subfolders called `ext` and `java`.
4. Copy the `JavaBridge.jar`, `php-script.jar`, and `script-api.jar` from the `JAVA.STANDALONE` folder into the `ext` folder.
5. Copy the `Java.inc` from the `JAVA.STANDALONE` folder into the `java` folder.
6. Put the `JavaBridge` folder in the same folder as the PHP files.

After the PHP/Java Bridge is installed, we need to revise the PHP file so that it can access the Java objects. To enable the PHP/Java Bridge in PHP, use the following command:

```
require_once("JavaBridge/java/Java.inc")
```

Before running the web application, the PHP/Java Bridge needs to be turned on. This can be done by double click the `JavaBridge.jar` file. A popup screen will ask the user to select a port for the servlet. Choosing the default port should be good enough. Then, the

web application can be started and the PHP file can access Java resource with PHP/Java Bridge. When installing the PHP/Java Bridge, the user should be extremely careful. These steps maybe slightly different based on the environment that the tool kit is installing to.

4.1.7 script.aculo.us

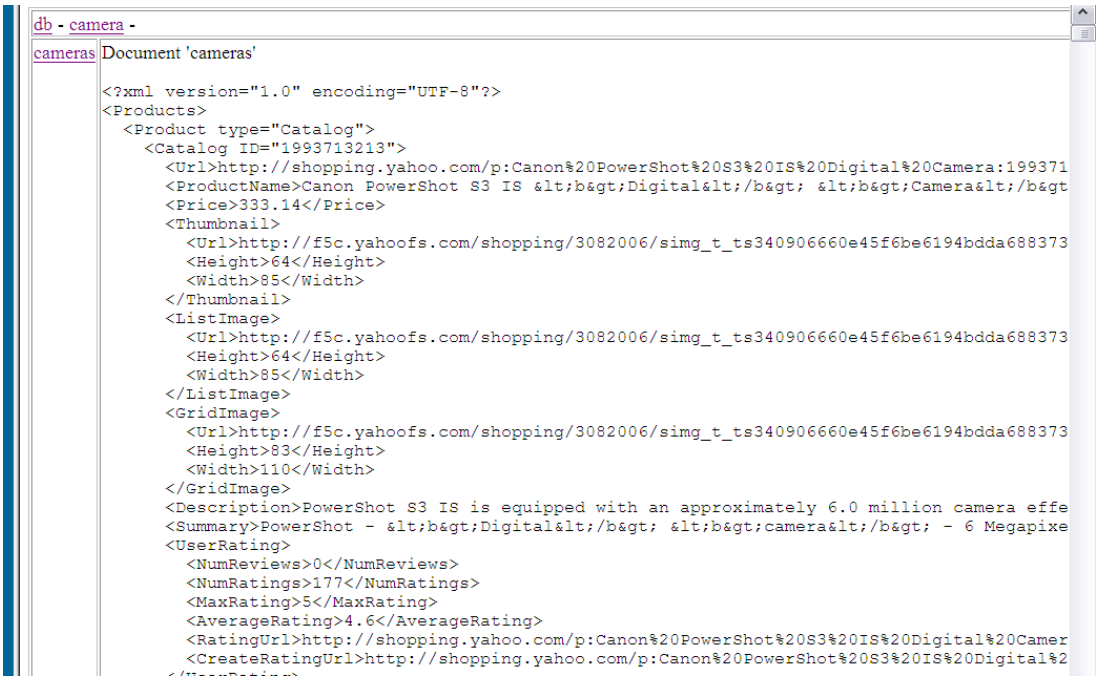
Since a slider interface is chosen in the design, an approach to implement a slider to accept the range input is needed. As a web application, the slider interface needs to be developed in JavaScript and it should contain one slide track and three slide bars. Currently, no such kind of slider tool kit is available on the Internet. Therefore, I use the script.aculo.us package to implement this slider. The script.aculo.us package is a JavaScript library which can provide dynamic visual effects and user interface elements to web application [14]. The package is built on the Prototype JavaScript Framework and it can also work with other web application framework and scripting languages. To use the package, extract the `prototype.js` and the `scriptaculous.js` from the downloaded file. Then load these two files in the web application as external JavaScript resource. In this implementation, I use this package with HTML and PHP to develop an intuitive user interface which can accept both range criteria and preference in one single slider.

4.2 Experimental Result

The implementation of the fuzzy range query design has been constantly improved during the development process. Here I present three demonstrations to illustrate the usability of the project. These demonstrations include one for fuzzy numerical range query, one for fuzzy linguistic range query, and one for combination of fuzzy numerical and linguistic range query.

4.2.1 Test Scenario

All demonstrations simulate a scenario which is an online product search service. The user will be able to search for digital camera based on range criteria. In the implementation, the two range criteria are the price of the digital camera and the image quality of the digital camera. The price of the digital camera is used for fuzzy numerical range query and the image quality of the digital camera is used for fuzzy linguistic range query. The sample data is collected from the Yahoo Shopping API [15]. This API can return product information in XML structure. In this case, the XML data returned by the Yahoo Shopping API is the product information on digital camera. This XML data is then stored in Apache Xindice as testing data. Figure 22 shows a portion of the testing data stored in Apache Xindice and the appendix lists all the price and image quality value for each camera.



db - camera -

cameras Document 'cameras'


```
<?xml version="1.0" encoding="UTF-8"?>
<Products>
  <Product type="Catalog">
    <Catalog ID="1993713213">
      <Url>http://shopping.yahoo.com/p:Canon%20PowerShot%20S3%20IS%20Digital%20Camera:199371
      <ProductName>Canon PowerShot S3 IS &lt;b&gt;Digital&lt;/b&gt; &lt;b&gt;Camera&lt;/b&gt
      <Price>333.14</Price>
      <Thumbnail>
        <Url>http://f5c.yahoofs.com/shopping/3082006/simg_t_ts340906660e45f6be6194bdda688373
        <Height>64</Height>
        <Width>85</Width>
      </Thumbnail>
      <ListImage>
        <Url>http://f5c.yahoofs.com/shopping/3082006/simg_t_ts340906660e45f6be6194bdda688373
        <Height>64</Height>
        <Width>85</Width>
      </ListImage>
      <GridImage>
        <Url>http://f5c.yahoofs.com/shopping/3082006/simg_t_ts340906660e45f6be6194bdda688373
        <Height>83</Height>
        <Width>110</Width>
      </GridImage>
      <Description>PowerShot S3 IS is equipped with an approximately 6.0 million camera effe
      <Summary>PowerShot - &lt;b&gt;Digital&lt;/b&gt; &lt;b&gt;camera&lt;/b&gt; - 6 Megapixe
      <UserRating>
        <NumReviews>0</NumReviews>
        <NumRatings>177</NumRatings>
        <MaxRating>5</MaxRating>
        <AverageRating>4.6</AverageRating>
        <RatingUrl>http://shopping.yahoo.com/p:Canon%20PowerShot%20S3%20IS%20Digital%20Camer
        <CreateRatingUrl>http://shopping.yahoo.com/p:Canon%20PowerShot%20S3%20IS%20Digital%2
```


Figure 22: Testing data in Apache Xindice

4.2.2 Fuzzy Numerical Range Query

Figure 23 shows the interface of the implementation for fuzzy numerical range query. The interface allows the user to specify a range on the price of the digital cameras he/she is looking for. As shown in the figure, the minimum price for the search range is $0 and the maximum price for the search range is $700. This means all digital cameras have a price between $0 and $700. The length of the slide track represents all possible prices available for searching. The upper bound slide bar on the right is illustrated with a left-pointing arrow. This slide bar is used for accepting the upper bound of the price range from the user. The slide bar can be dragged along the slide track but it cannot pass other slide bars as defined in the design. A popup balloon is shown along the slide bar when it is being point to. The popup balloon not only helps the user to easily identify which value he/she is adjusting, but also indicates the placement of the slide bar on the slide track. In this example, the upper bound slide bar is located at about 43% of the full length of the slide track. The other slide bars are the price lower bounder slide bar shown as a right-pointing arrow and the preferred price slide bar shown as an up-pointing arrow. The input range is indicated with different color on the slide track. Once all three slide bars are set to the expected place, the user can press the "Submit" button to send the input to the search engine.

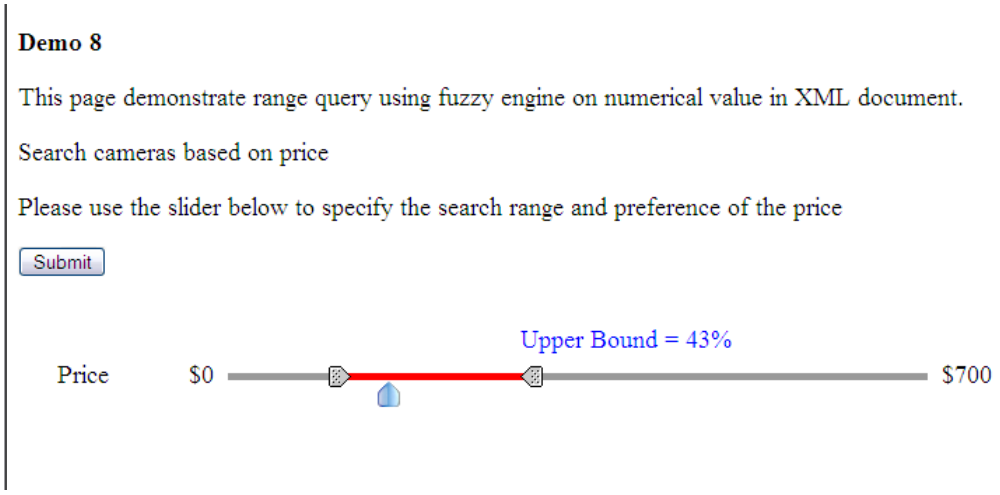


Figure 23: Fuzzy numerical range query interface

The fuzzy search engine then takes the user input to calculate the score for each camera and output the search result on the screen as shown in Figure 24. The output first shows the input values from the user. When the slide bars are set to be in the place as shown in Figure 23, the price lower bound is set to be 0.2231, the preferred price is set to be 0.2955 and the price upper bound is set to be 0.4339. Since the input from the slider is always set to be from 0 to 1. These slide bar values need to be mapped to price value. The prices for lower bound, preferred price, and upper bound are $156.20, $206.82, and $303.72 respectively.

User input:
Price Lower Bound: 0.2231 ($156.20)
Preferred Pirce: 0.2955 ($206.82)
Price Upper Bound: 0.4339 ($303.72)

Search Result:		
Price	Product Name	Score
$208.99	Kodak EasyShare Z740 **Digital Camera**	91.6115
$198.99	Canon PowerShot A540 **Digital Camera**	91.3497
$198.46	Kodak EasyShare C300 **Digital Camera**	91.3098
$179.99	Sony Cyber-shot DSC-W55/P **Digital Camera**	88.2261
$238.68	Casio EXILIM EX-Z1200 **Digital Camera**	86.8410
$238.99	Canon PowerShot SD600 **Digital Camera**	86.7494
$169.95	Sony Cyber-shot DSC-W80 **Digital Camera**	85.2905
$245.00	Sony Cyber-shot DSC-W200 **Digital Camera**	84.8629
$249.10	Sony Cyber-shot DSC-T9 **Digital Camera**	83.4969
$155.95	Canon PowerShot SD750 **Digital** ELPH **Digital Camera**	79.2221
$289.00	Canon PowerShot S5 IS **Digital Camera**	75.0702
$299.00	Canon PowerShot A630 **Digital Camera**	73.4539
$299.34	Olympus Stylus 710 **Digital Camera**	73.3474
$149.99	Sony Cyber-shot DSC-W55 **Digital Camera**	45.4058
$149.95	Kodak EasyShare C875 **Digital Camera**	45.2285
$149.77	Casio EXILIM ZOOM EX-Z75 **Digital Camera**	44.4410
$149.00	Canon PowerShot SD1000 **DIGITAL** ELPH (1862B001) **Digital Camera**	41.2673
$142.99	Canon PowerShot A550 **Digital Camera**	33.8331
$139.00	Canon PowerShot A410 **Digital Camera**	32.0072
Total 19 cameras found		

Figure 24: Fuzzy numerical range query result

Now let's examine the search result. Since the preferred price is set to be $206.82, digital cameras with a price close to $206.82 should show up first. In the search result, the Kodak EasyShare Z740 is the first digital camera returned since it has a price of $208.99. Also, the Sony Cyber-shot DSC-W55/P at $179.99 is listed before the Casio EXILIM EX-Z1200 at $238.68 since its price is closer to the preferred price. The price range of

the input is from $156.20 to $303.72. Several digital cameras with price lower than but still close to the lower bound price are shown because of the fuzzy algorithm. On the other hand, no camera with price higher than the upper bound price is shown. This is because the first camera has a price higher than $303.72 is the Canon PowerShot S3 IS at $333.14. This price is greater than the upper bound price way too much so that the camera is not included in the search result by the fuzzy search engine. From this example, we can see the fuzzy numerical range query performs as expected.

4.2.3 Fuzzy Linguistic Range Query

The implementation for the fuzzy linguistic range query is based on one for the fuzzy numerical range query. This implementation allows users to search for digital camera based on the image quality. The interface, as shown in Figure 25, is slightly changed to meet the needs for linguistic range query. The slide track now represents all possible value for the image quality of the digital cameras. The minimum image quality for the digital camera is “Poor” and the maximum image quality for the digital camera is “Excellent”. The user can use the slide bars to set the range of the image quality and the preferred image quality. Notice the slide bars implicitly represent the linguistic value of the image quality. In the example, the lower bound slide bar is set to be about half way between “Poor” and “Excellent”. The linguistic value which has a meaning “half way between poor and excellent” can be “Normal”, “Average”, “Common”, and so on. Which linguistic value is being actually stored in the database does not matter to the interface. That is because as long as the linguistic value has a semantic of “half way between poor and excellent”, it can be considered as the user expected value. This way, the user doesn’t need to know the exact value of the linguistic term. He/she can simple set the input value based on his/her own understanding.

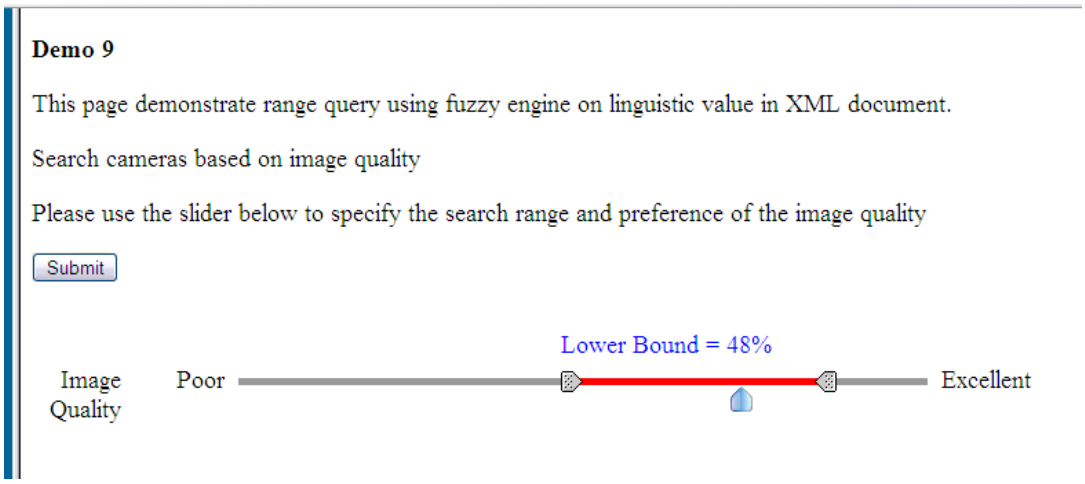


Figure 25: Fuzzy linguistic range query interface

After the input from the user is accepted, the search result is displayed on the screen as shown in Figure 26. The input is 0.4835 for the image quality lower bound, 0.7355 for the preferred image quality, and 0.8636 for the image quality upper bound. The image quality lower bound has a 6.6% weight on the linguistic value “Bad” and a 93.4% weight on the linguistic value “Average”. This means the image quality lower bound is very close to the linguistic value “Average”. The weight is calculated based on the hamming distance between the input value and the fuzzified value of the linguistic term [16]. Same applies to the preferred image quality and the image quality upper bound. From the figure, the preferred image quality has a 5.8% weight on the linguistic value “Average” and a 94.2% weight on the linguistic value “Good”. This concludes that the preferred image quality is very close to be “Good”. The weight for the image quality upper bound is 54.5% on “Good” and 45.5% on “Excellent”. Therefore, the upper bound of the image quality is set to be “between Good and Excellent”.

User input:
Image Quality Lower Bound: 0.4835 (6.6%Bad, 93.4%Average)
Preferred Image Quality: 0.7355 (5.8%Average, 94.2%Good)
Image Quality Upper Bound: 0.8636 (54.5%Good, 45.5%Excellent)

Search Result:		
Image Quality	Product Name	Score
Good	Canon PowerShot S5 IS **Digital Camera**	99.4000
Good	Kodak EasyShare C300 **Digital Camera**	99.4000
Good	Panasonic Lumix DMC-FZ50-K (Black) **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-W7 **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-W200 **Digital Camera**	99.4000
Good	Casio EXILIM EX-Z1200 **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-H9/B **Digital Camera**	99.4000
Good	Nikon D300 **Digital Camera**	99.4000
Good	Canon EOS **Digital** Rebel XTi (Black) **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-T100 **Digital Camera**	99.4000
Good	Kodak EasyShare Z740 **Digital Camera**	99.4000
Good	Fujifilm FinePix S9000 **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-R1 **Digital Camera**	99.4000
Good	Sony Cyber-shot DSC-N1 **Digital Camera**	99.4000
Average	Sony Cyber-shot DSC-S700 **Digital Camera**	17.8000
Average	Samsung S630 **Digital Camera**	17.8000
Average	Nikon D40 **Digital Camera**	17.8000
Average	Canon PowerShot SD600 **Digital Camera**	17.8000
Average	VisionTek Argus QuickClix 3185 **Digital Camera**	17.8000
Average	Canon PowerShot A630 **Digital Camera**	17.8000
Average	Canon EOS 5D **Digital Camera**	17.8000
Average	Canon PowerShot G9 **Digital Camera**	17.8000
Average	Canon PowerShot S3 IS **Digital Camera**	17.8000
Average	Sony Cyber-shot DSC-W55 **Digital Camera**	17.8000
Average	Sony Cyber-shot DSC-S650 **Digital Camera**	17.8000
Average	Fujifilm FinePix A610 **Digital Camera**	17.8000
Average	Olympus Stylus 710 **Digital Camera**	17.8000
Average	Canon PowerShot A410 **Digital Camera**	17.8000
Average	Canon EOS **Digital** Rebel XTi (Body Only-Black) **Digital Camera**	17.8000
Total 29 cameras found		

Figure 26: Fuzzy linguistic range query result

Since the input range of the image quality is set to be between “Average” and somewhere “between Good and Excellent”. The search result returns digital cameras with “Good” and “Average” image quality. Digital cameras with “Good” image quality show up first because the preferred image quality is set to be “Good”. Notice that digital cameras with the same quality have the same score because same linguistic value is fuzzified into same membership degree. Digital cameras with “Excellent” image quality are not included in the search result is because the image quality upper bound is “between Good and Excellent”. The membership degree of “Excellent” is a little bit too far from the image quality upper bound therefore it cannot be picked up by the fuzzy search engine. Based on the search result, the fuzzy linguistic range query can return the expected result.

4.2.4 Combination of Fuzzy Numerical and Linguistic Range Query

An implementation which can handle both fuzzy numerical and linguistic range query is done by combining the two implementations above. This implementation allows the user search for digital camera based on both price and image quality. The slider interface is still used to accept user input. The only difference is that two slider are used as shown in Figure 27 because two search criteria are given.

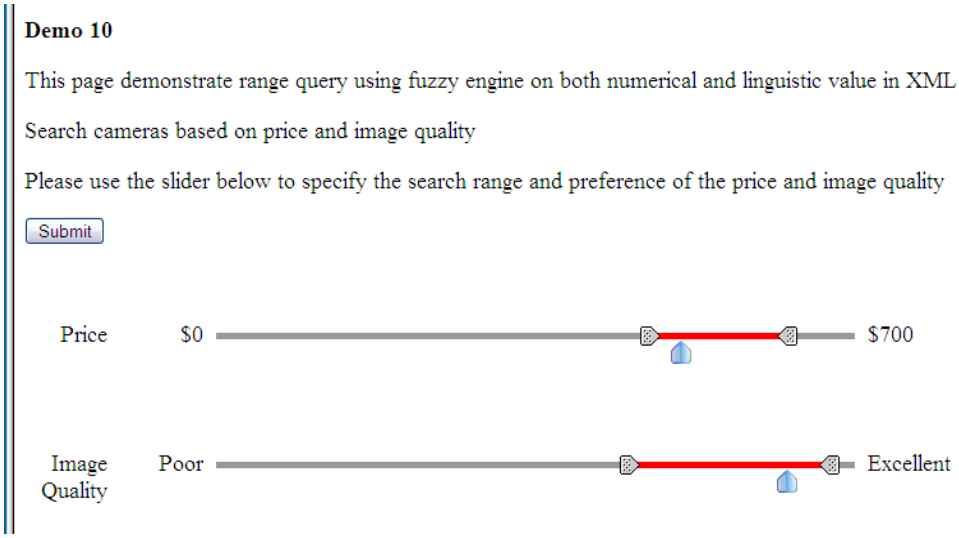


Figure 27: Fuzzy numerical and linguistic range query interface

Just as the other implementation, the user input is displayed first for examination. In Figure 28, the user set the price lower bound, the preferred price, and the price upper bound to be 0.6860, 0.7335, and 0.9070. By mapping these input values to available price range, we get $480.17 for the price lower bound, $513.43 for the preferred price, and $634.92 for the price upper bound. For the image quality, the lower bound is 0.6529 with a 38.8% weight on “Average” and a 61.2% weight on “Good”, the preference is 0.9050 with a 38% weight on “Good” and a 62% weight on “Excellent”, and the upper bound is 0.9752 with a 9.9% weight on “Good” and a 90.1% weight on “Excellent”. This means the image quality lower bound is “not so Good but better than Average”. The preferred image quality is “not so Excellent but better than Good”. The image quality upper bound is “almost Excellent”. Therefore, the user prefers a digital camera with a price of $513.43 and a “not so Excellent but better than Good” image quality.

The first digital camera returned is the Canon EOS 40D which has a price of $528.00 and an “Excellent” image quality. This is even better than the user preferred since its image quality is better than the user expected and its price is only slightly higher than the user expected. The second and third returned digital cameras are the Nikon D80 (Body Only) and the Sony Cyber-shot DSC-R1. The Nikon D80 (Body Only) is listed before the Sony Cyber-shot DSC-R1 because it has a better image quality than the later one. Although its price is further from the user preferred price, the image quality outweighs the price because the difference between image qualities is larger than the difference between prices. Also notice that Canon EOS 5D with a price at $519.00 and image quality of “Average” shows up in the result. This is because its price is very close to the preferred price, which is $513.43, and its image quality is not too far from the preferred image quality, which is “not so Good but better than Average”. At last, some cameras with price outside of the user specified price range are included because of their image quality fall within the user specified image quality range. Overall, the search result shows that the implementation can handle fuzzy numerical and linguistic range query very well at the same time.

User input:
Price Lower Bound: 0.6860 ($480.17)
Preferred Pirce: 0.7335 ($513.43)
Price Upper Bound: 0.9070 ($634.92)
Image Quality Lower Bound: 0.6529 (38.8%Average, 61.2%Good)
Preferred Image Quality: 0.9050 (38.0%Good, 62.0%Excellent)
Image Quality Upper Bound: 0.9752 (9.9%Good, 90.1%Excellent)

Search Result:					
Product Name	Price	Price Score	Image Quality	Image Quality Score	Total Score
Canon EOS 40D **Digital Camera**	$528.00	90.7333	Excellent	50.5000	**141.2333**
Nikon D80 (Body Only) **Digital Camera**	$585.00	76.3785	Excellent	50.5000	**126.8785**
Sony Cyber-shot DSC-R1 **Digital Camera**	$499.99	90.8687	Good	35.4000	**126.2687**
Canon EOS **Digital** Rebel XTi (Black) **Digital Camera**	$497.00	90.4858	Good	35.4000	**125.8858**
Sony Cyber-shot DSC-W7 **Digital Camera**	$479.00	78.8217	Good	35.4000	**114.2217**
Sony Cyber-shot DSC-N1 **Digital Camera**	$599.99	75.0600	Good	35.4000	**110.4600**
Nikon D300 **Digital Camera**	$629.00	67.0483	Good	35.4000	**102.4483**
Canon EOS 5D **Digital Camera**	$519.00	91.5031	Average	0.0000	**91.5031**
Kodak EasyShare C643 **Digital Camera**	$109.88	0.0000	Excellent	50.5000	**50.5000**
Sony Cyber-Shot DSC-H1 **Digital Camera**	$389.99	0.0000	Excellent	50.5000	**50.5000**
Casio EXILIM ZOOM EX-Z75 **Digital Camera**	$149.77	0.0000	Excellent	50.5000	**50.5000**
Nikon Coolpix S4 **Digital Camera**	$333.15	0.0000	Excellent	50.5000	**50.5000**
Canon PowerShot A540 **Digital Camera**	$198.99	0.0000	Excellent	50.5000	**50.5000**
Kodak EasyShare C530 **Digital Camera**	$89.95	0.0000	Excellent	50.5000	**50.5000**
Canon PowerShot S5 IS **Digital Camera**	$289.00	0.0000	Good	35.4000	**35.4000**
Kodak EasyShare C300 **Digital Camera**	$198.46	0.0000	Good	35.4000	**35.4000**
Sony Cyber-shot DSC-T100 **Digital Camera**	$399.99	0.0000	Good	35.4000	**35.4000**
Sony Cyber-shot DSC-H9/B **Digital Camera**	$340.00	0.0000	Good	35.4000	**35.4000**
Panasonic Lumix DMC-FZ50-K (Black) **Digital Camera**	$453.00	0.0000	Good	35.4000	**35.4000**
Casio EXILIM EX-Z1200 **Digital Camera**	$238.68	0.0000	Good	35.4000	**35.4000**
Sony Cyber-shot DSC-W200 **Digital Camera**	$245.00	0.0000	Good	35.4000	**35.4000**
Fujifilm FinePix S9000 **Digital Camera**	$349.00	0.0000	Good	35.4000	**35.4000**
Kodak EasyShare Z740 **Digital Camera**	$208.99	0.0000	Good	35.4000	**35.4000**
Total 23 cameras found					

Figure 28: Fuzzy numerical and linguistic range query result

5 Conclusion and Future Enhancement

5.1 Conclusion

Current solutions on range query have various limitations. This project introduced a new solution to perform range query against XML data. Demonstrations have shown this new solution can fuzzify the user specified range so that more meaningful search result can be returned. The solution also allows the user to specify preference value which provides more control over the search result to the user. Besides these, the solution is flexible enough to handle fuzzy range query on either numerical or linguistic values. Even combination of both numerical and linguistic fuzzy range query can be performed with this solution.

Again, the new solution is not designed to replace the current solutions. For situations such as exact range is needed or user preference can be assumed, the new solution may not be necessary. Therefore, developers should choose to use the new solution or the current solution based on the application needs.

After all, this project provided an alternative approach for performing range query over XML data. The combination of fuzzy logic and search engine, the innovation on interface, and the utilization on native XML database together form an intelligent solution for the expanding online search technology.

5.2 Future Enhancement

The current implementation retrieves all items from the database and then performs an evaluation on the items with the search criteria to calculate the score value. This causes high traffic volume between the search engine and the database. An improvement should be made so that only items tend to match the search criteria will be passed to the search engine.

Allowing user specified weight on the search criteria is also an interesting enhancement. Different search criteria are treated fairly right now. It will be more user-friendly if the user can indicate one search criteria is more important than others.

6 Reference

[1] H. L. Kwang, and J. H. Lee. “A Method for Ranking Fuzzy Numbers and Its Application to Decision-Making,” in *IEEE Transaction on Fuzzy Systems*, 1999, pp. 667-685.

[2] B. K. Mohanty, and K. Passi. “Web based information for product ranking in e-business: a fuzzy approach,” in ACM International Conference Proceeding Series, New York, 2006, Vol. 156, pp. 558-563.

[3] J. McGrenere, R. M. Baecker, and K. S. Booth. “A Field Evaluation of an Adoptable Two-Interface Design for Feature-Rich Software,” in *ACM Transaction on Computer-Human Interaction*, New York, 2007, Vol. 14.

[4] D. Tao, and C. Tseng. “Fuzzy Logic and its applications,” CS274 class material.

[5] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. “Searching XML Documents via XML Fragments,” in *Annual ACM Conference on Research and Development in Information Retrieval*, Toronto, 2003, pp. 151-158.

[6] D. Che, K. Aberer, and T. Ozsu. “Query optimization in XML structured-document databases,” in *The International Journal of on Very Large Data Bases*, New York, 2006, Vol. 15, pp. 263-289.

[7] G. Xing, J. Esanakula, and S. Jayanty. “Index and Storage Design in Native XML Databases,” in *ACM Southeast Regional Conference*, New York, 2005, Vol. 1, pp. 218-219.

[8] Fuzzy Control Language. http://jfuzzylogic.sourceforge.net/doc/iec_1131_7_cd1.pdf.

[9] jFuzzyLogic project. http://jfuzzylogic.sourceforge.net/.

[10] Apache Friends Project XAMPP. http://www.apachefriends.org/en/xampp.html.

[11] Apache Xindice project. http://xml.apache.org/xindice/.

[12] Apache Tomcat project. http://tomcat.apache.org/.

[13] PHP/Java Bridge project. http://php-java-bridge.sourceforge.net/.

[14] The script.aculo.us project. http://script.aculo.us/.

[15] Yahoo Developer Network. http://developer.yahoo.com/.

[16] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. “Comparing and aggregating rankings with ties,” in *Symposium on Principle of Database System*, Paris, 2004, pp. 47-58.

7 Appendix: Search Data Pool

Product Name	Price	Image Quality
VisionTek Argus QuickClix 3185 Digital Camera	$35.05	Average
Fujifilm FinePix A610 Digital Camera	$62.91	Average
Samsung S630 Digital Camera	$79.72	Average
Nikon Coolpix L11 Digital Camera	$79.99	Bad
GE E1030 Digital Camera	$87.99	Poor
Sanyo VPC-T700 Digital Camera	$89.00	Bad
Kodak EasyShare C530 Digital Camera	$89.95	Excellent
Kodak EasyShare C643 Digital Camera	$109.88	Excellent
Sony Cyber-shot DSC-S650 Digital Camera	$119.90	Average
Nikon Coolpix 5600 Digital Camera	$119.95	Poor
Sony Cyber-shot DSC-S700 Digital Camera	$119.99	Average
Canon PowerShot A410 Digital Camera	$139.00	Average
Canon PowerShot A550 Digital Camera	$142.99	Poor
Canon PowerShot SD1000 DIGITAL ELPH (1862B001) Digital Camera	$149.00	Bad
Casio EXILIM ZOOM EX-Z75 Digital Camera	$149.77	Excellent
Kodak EasyShare C875 Digital Camera	$149.95	Poor
Sony Cyber-shot DSC-W55 Digital Camera	$149.99	Average
Canon PowerShot SD750 Digital ELPH Digital Camera	$155.95	Bad
Sony Cyber-shot DSC-W80 Digital Camera	$169.95	Poor
Sony Cyber-shot DSC-W55/P Digital Camera	$179.99	Bad
Kodak EasyShare C300 Digital Camera	$198.46	Good
Canon PowerShot A540 Digital Camera	$198.99	Excellent
Kodak EasyShare Z740 Digital Camera	$208.99	Good
Casio EXILIM EX-Z1200 Digital Camera	$238.68	Good
Canon PowerShot SD600 Digital Camera	$238.99	Average
Sony Cyber-shot DSC-W200 Digital Camera	$245.00	Good
Sony Cyber-shot DSC-T9 Digital Camera	$249.10	Bad
Canon PowerShot S5 IS Digital Camera	$289.00	Good
Canon PowerShot A630 Digital Camera	$299.00	Average
Olympus Stylus 710 Digital Camera	$299.34	Average
Canon PowerShot S3 IS Digital Camera	$333.14	Average
Nikon Coolpix S4 Digital Camera	$333.15	Excellent
Sony Cyber-shot DSC-H9/B Digital Camera	$340.00	Good
Fujifilm FinePix S9000 Digital Camera	$349.00	Good

Canon PowerShot A520 Digital Camera	$384.28	Bad
Sony Cyber-Shot DSC-H1 Digital Camera	$389.99	Excellent
Nikon D40 Digital Camera	$392.00	Average
Sony Cyber-shot DSC-T100 Digital Camera	$399.99	Good
Canon PowerShot G9 Digital Camera	$416.00	Average
Canon EOS Digital Rebel XTi (Body Only-Black) Digital Camera	$429.00	Average
Panasonic Lumix DMC-FZ50-K (Black) Digital Camera	$453.00	Good
Sony Cyber-shot DSC-W7 Digital Camera	$479.00	Good
Canon EOS Digital Rebel XTi (Black) Digital Camera	$497.00	Good
Sony Cyber-shot DSC-R1 Digital Camera	$499.99	Good
Canon EOS 5D Digital Camera	$519.00	Average
Canon EOS 40D Digital Camera	$528.00	Excellent
Nikon D80 (Body Only) Digital Camera	$585.00	Excellent
Sony Cyber-shot DSC-N1 Digital Camera	$599.99	Good
Nikon D300 Digital Camera	$629.00	Good
Nikon D80 Digital Camera	$674.00	Bad