

ITCS 6155 – KNOWLEDGE BASED SYSTEMS

FLEXIBLE QUERY ANSWERING SYSTEM (FQAS)

PROJECT SUBMITTED BY,

STUDENT ID	NAME	EMAIL
800781950	Jude Nishanth Wilfred	jwilfred@uncc.edu
800765188	Guhan Murugesan	gmuruges@uncc.edu
800767474	Mohamed Zakriea Niyaz Mohamed Rafick	mmohame4@uncc.edu

TABLE OF CONTENTS

S. No.	TOPIC	PAGE
1	About the Project	3
2	Data Preparation	3
3	Implementation Steps	4
	Extracting Domain Knowledge	4
	Finding the “Refiner” Statement	5
	Refining the Failing Conjunction	6
4	Steps for Running the Project	7
5	References	8

1. ABOUT THE PROJECT

The Flexible Query Answering System (FQAS) provides a solution to the Failing Query Problem: given a query that returns an empty answer, how can one relax the query's constraints so that it returns a non-empty set of tuples? The FQAS relaxes the queries that are in disjunctive normal form so that some non-empty result is returned.

The Music Dataset contains 323 attributes for each music file. The objective of the project is to process the user queries, relax them if they are failing queries and return the results that most closely match the given query. The user queries are in disjunctive normal form.

The project uses Muslea's Algorithm presented in the paper **Machine Learning for Online Query Relaxation**.

2. DATA PREPARATION

The given dataset is in .xlsx (Microsoft Excel) format. This format is not accepted by WEKA. So, the dataset was converted to .csv (Comma Separated Values) format. Also, there were several non-numeric values which cannot be handled by both WEKA and Java. Hence, all NaN (Not a Number) values were replaced with 0 (zero).

Three text files namely **Columns.txt**, **Maximum.txt** and **Minimum.txt** were created (In Data folder). The contents of each of these files are listed as follows:

- **Columns.txt**: Contains the list of attribute names as given in the dataset
- **Maximum.txt**: Contains the maximum values for each attribute in the dataset
- **Minimum.txt**: Contains the minimum values for each attribute in the dataset

The contents of each of these text files are stored in 3 different HashMaps and their elements are retrieved using the get() method of the HashMaps when required.

3. IMPLEMENTATION STEPS

1. Extracting Domain Knowledge:

Based on the user query, the original dataset is queried to project a subset of the original dataset that contains only the attributes specified in the user query. Using **MetaModel API** we fetch the projected data. For each constraint in the user query, the values in the associated column are tested using **Math package** and the values are replaced with YES if the values satisfy the constraint or NO otherwise. The modified datasets for each attribute are written to different **.csv files**.

In order to apply the **C4.5 algorithm**, WEKA requires the input file to be in **.arff** (Attribute Relation File Format) format. So each of the .csv files are read and are converted to .arff files. Then, those .arff files are read with WEKA's **J48 Class** which is the implementation of C4.5 algorithm.

	A	B	C	D	E
1	spectral_mfcc	spectral_mfcc_Me	spectral_mfcc_St	spectral_mfcc	spectral_mfcc_Std_3
2	NO	-0.021515	0.770673	0.556085	0.782519
3	NO	-0.026606	0.268881	0.138238	0.140945
4	NO	0.139152	0.247422	0.19677	0.348365
5	NO	0.0264	0.407016	0.221909	0.231531
6	YES	-0.242507	0.169761	0.213729	0.147867
7	NO	-0.230754	0.236674	0.356777	0.300458
8	NO	-0.126732	0.412882	0.203045	0.346893
9	NO	-0.115622	0.162608	0.159445	0.156304
10	NO	-0.130842	0.2954	0.138899	0.139438
11	NO	-0.017656	0.228644	0.138953	0.170542
12	NO	-0.051843	0.154279	0.137218	0.151665
13	NO	0.022912	0.448981	0.158166	0.244389
14	NO	0.022213	0.61057	0.160726	0.228595
15	NO	-0.462218	0.382945	0.12466	0.252541

Disk (D:) > MyWorkspace > KBS_PROJ > Temp_Files					Search Temp_Files
e in library > Share with > Burn > New folder					
Name	Date modified	Type	Size		
spectral_mfcc_Mean_12	5/2/2013 2:39 PM	ARFF Data File	57 KB		
spectral_mfcc_Mean_12	5/2/2013 2:39 PM	Microsoft Excel C...	57 KB		
spectral_mfcc_Mean_13	5/2/2013 2:39 PM	ARFF Data File	57 KB		
spectral_mfcc_Mean_13	5/2/2013 2:39 PM	Microsoft Excel C...	57 KB		
spectral_mfcc_Std_1	5/2/2013 2:39 PM	ARFF Data File	57 KB		
spectral_mfcc_Std_1	5/2/2013 2:39 PM	Microsoft Excel C...	57 KB		
spectral_mfcc_Std_2	5/2/2013 2:39 PM	ARFF Data File	57 KB		
spectral_mfcc_Std_2	5/2/2013 2:39 PM	Microsoft Excel C...	57 KB		
spectral_mfcc_Std_3	5/2/2013 2:39 PM	ARFF Data File	58 KB		
spectral_mfcc_Std_3	5/2/2013 2:39 PM	Microsoft Excel C...	58 KB		

The C4.5 algorithm results in a decision tree. In order to obtain the rules from that decision tree, we use the **PART Class** of WEKA. PART Class provides a summary of all the rules. So, to get the rules, a private member of PART Class called **m_root**, a **MakeDecList** object, was made public. Also, a method to get the **rules vector**, **getRules()**, is added to WEKA and compiled to a JAR.

In PART Class

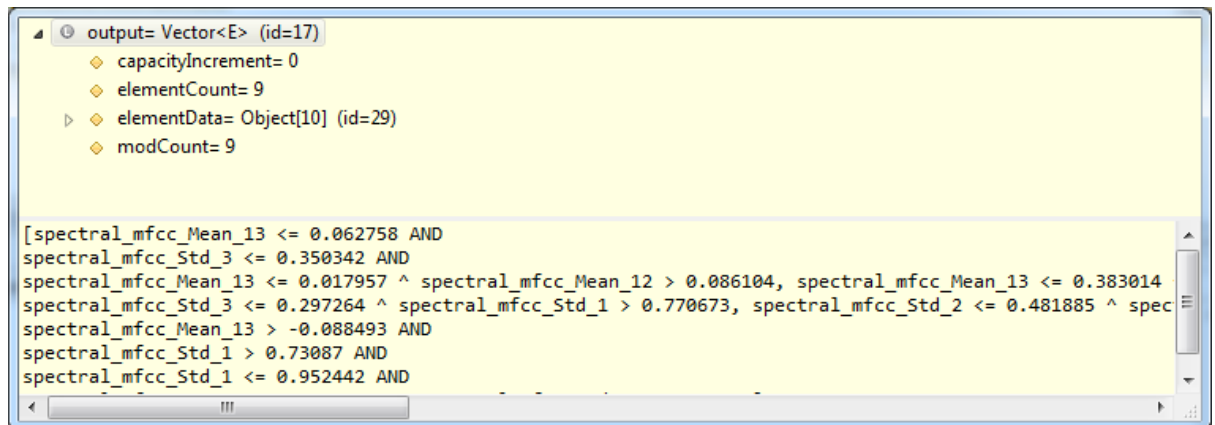
```
/** The decision list */ // Modified - jude
public MakeDecList m_root;
```

In MakeDecList

```
* Returns the Rules -- modified - Jude
*/

public Vector getRules() {
    return theRules;
}
```

Rule Statements Obtained



2. Finding the “refiner” statement

The input query entered by the user will be of the following format:

$$A_1 \geq V_1 \wedge A_2 \leq V_2 \wedge \dots \wedge A_n > V_n$$

where, A_1, A_2, \dots, A_n are attribute names present in the given dataset.

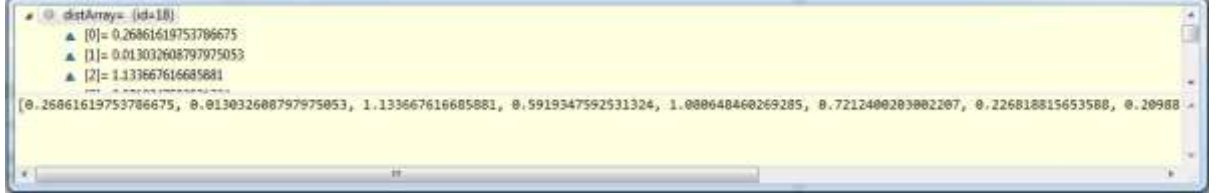
The input query is divided into 3 arrays:

- **QueryString[]**: Array that contains the list of attributes - A_1, A_2, \dots, A_n
- **op[]**: Array that contains the list of constraints
- **arg[]**: Array that contains the list of values - V_1, V_2, \dots, V_n

The elements of QueryString[] are compared with each element of the rules vector. The distances between each rule in the rules vector and the input query are calculated according to the following formula:

$$Dist(C_k, S_i, A_j) = \frac{|Value(Constr_{C_k}(A_j)) - Value(Constr_S(A_j))|}{Max_{A_j} - Min_{A_j}}$$

These distances are stored in distArray[] as follows:



The value of distance which is smallest in distArray[] is determined and hence the corresponding statement associated with that distance is extracted from the rules vector.

3. Refining the failing conjunction

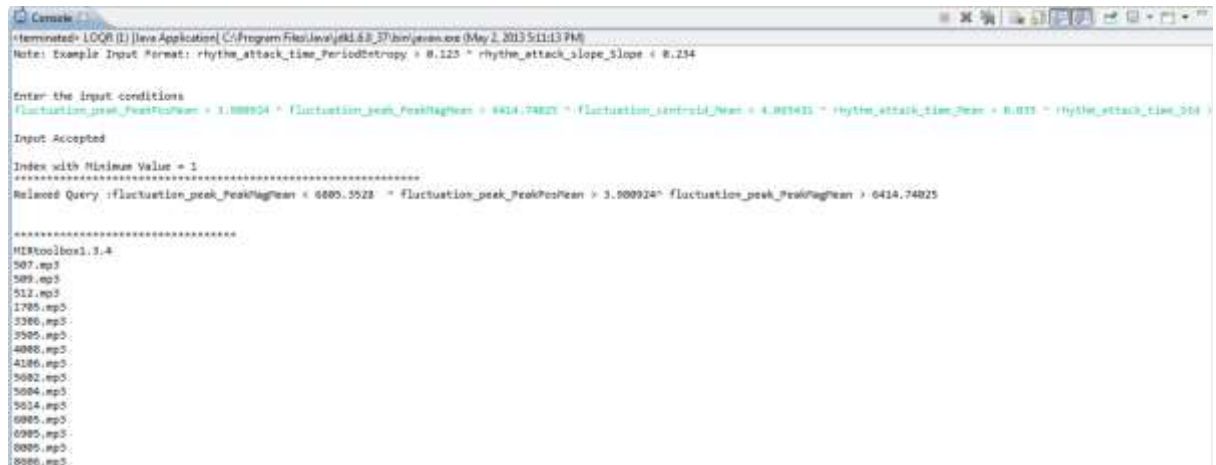
The relaxed conjunction includes only constraints on attributes that are present in both the user input and the best statement obtained from the previous step. Since the data contains only continuous values, we are considering only two possible scenarios:

- If the two constraints are of same type inequality, then we choose the least constraining of both constructs
- If the two constraints are of different inequalities, then we choose the intersection of the two constraints. If the result of intersection is empty, then there will be no changes to the relaxed conjunction

Once the conjunctions are relaxed, the music file names listed in the first column of the data set which satisfy the conjunctions are fetched and displayed as the final result of the user's input query.

Sample Output:

Input Query: $fluctuation_peak_PeakPosMean > 3.980924 \wedge$
 $fluctuation_peak_PeakMagMean > 6414.74025 \wedge fluctuation_centroid_Mean <$
 $4.065431 \wedge rhythm_attack_time_Mean > 0.035 \wedge rhythm_attack_time_Std > 0.017321$



```

*terminated* LOQR (1) [Java Application] C:\Program Files\Java\jdk1.6.0_37\bin\javaw.exe (May 2, 2013 5:11:13 PM)
Notes: Example Input Format: rhythm_attack_time_PeriodEntropy > 0.125 ^ rhythm_attack_slope_slope < 0.234

Enter the input conditions
fluctuation_peak_PeakPosMean > 3.980924 ^ fluctuation_peak_PeakMagMean > 6414.74025 ^ fluctuation_centroid_Mean < 4.065431 ^ rhythm_attack_time_Mean > 0.035 ^ rhythm_attack_time_Std > 0.017321

Input Accepted

Index with Minimum Value = 1
*****
Relaxed Query :fluctuation_peak_PeakMagMean < 6805.3528 ^ fluctuation_peak_PeakPosMean > 3.980924 ^ fluctuation_peak_PeakMagMean > 6414.74025
*****

H:\Tools\box3.3.4
507.mp3
509.mp3
512.mp3
1795.mp3
3300.mp3
3505.mp3
4000.mp3
4100.mp3
5002.mp3
5004.mp3
5014.mp3
6005.mp3
6005.mp3
6005.mp3
6006.mp3

```

The Input Query given above is a failing query and so it will not produce any results. The FAQS relaxes the query as follows:

$fluctuation_peak_PeakMagMean < 6805.3528 \wedge fluctuation_peak_PeakPosMean >$
 $3.980924 \wedge fluctuation_peak_PeakMagMean > 6414.74025$

The FQAS then finds the music filenames that satisfy the above relaxed query and displays the results.

4. STEPS FOR RUNNING THE PROJECT

1. Extract KBS_PROJ.zip
2. Import the KBS_PROJ folder to Eclipse
3. Configure Build Path to add JARs to the project from the lob folder
4. Run LOQR.java
5. Enter the input query in the format exactly as specified in the console
6. The output generated shows:
 - Index of the best conjunction from the rule vector (just to show the output of the step)
 - Relaxed conjunction
 - Music Filenames that satisfy the relaxed conjunction

5. REFERENCES

1. Converting CSV to ARFF
<http://weka.wikispaces.com/Converting+CSV+to+ARFF>
2. Using WEKA in Java
<http://weka.wikispaces.com/Use+WEKA+in+your+Java+code>
3. ANT
<http://weka.wikispaces.com/ANT>
4. Querying CSV File
<http://kasper.eobjects.org/2008/11/querying-csv-file.html>

Discussed the understandings of the algorithm with Prakash Venigalla