### [COMS E6111 Advanced Database Systems](http://www.cs.columbia.edu/~gravano/cs6111/index.html) Fall 2015

**Project 3**

1. **Name and UNI**

Bahul Jain - bkj2111

Parth Parekh - prp2121

1. **A list of all the files that you are submitting**

run.sh - Shell script which calls run.py python script.

run.py - Main Python Script which runs the project.

Integrated-Dataset.csv – The dataset that contains the all the relevant columns from which we extract associations rules.

Legend.csv – Contains the legend for all the codes in the dataset.

test.db – The sqlite3 database that we have used to operate on the dataset

1. **A clear description of how to run your program**

To run the program on clic machine, run the run.sh file using the command:

/home/bkj2111/ADBS/Project3/run.sh

If you are running the code from local computer, run the following command from the project directory:

python run.py

**Note**: All the parameters are taken in the command line after the program starts executing.

1. **Choice of Dataset**

Link: <https://data.cityofnewyork.us/Housing-Development/Complaint-Problems/a2nx-4u46>

Complaints received from building residents are one of the most common issues that a building management has to deal with. Deciding what kind of resources, and employees are required to maintain the condition of buildings and to address to most frequently occurring problems in such households. The dataset we chose contains a list of complaints regarding problems that happen in such households. Each of the complaints contain three important fields that can be used to extract some valuable information :

1. Space Type: Describes the area in the house where problem happens.
2. Type: The urgency of the problem.
3. Major Category: A high level description of the problem.

Finding associations among these attributes could help building management companies and landlords address some of the most frequently occurring household maintenance issues in more informed and efficient ways.

1. **A clear description of the internal design of your project**

The program takes 4 inputs:

1. **The Bing Account Key**
2. **Database Name** - This is the database we are going to classify and prepare a content summary of based on the classification.
3. **Coverage Threshold** - This is minimum number of documents associated with a particular category that should be present in the database for the database to be considered for classification under that category.
4. **Specificity Threshold** - This is minimum percentage of the documents associated with a particular category that should be present in the database for the database to be considered for classification under that category.

Once these parameters are taken, using the Bing's Search API we query a set of keywords on the database. This is given to us already. We first start with all the keywords in the list 'Root.pickle'

**Note:** We have already created pickle files for every keyword list. These pickle files essentially store the exact same list in a default dictionary form. When we import the pickle file it directly gets us a dictionary is python usable form. The pickle file after importing looks something like this -

So for Category = 'Computers' we have stored all its respective query words in one list. Similarly for other categories query words are stored in this format.

While querying the database for a particular query word, we collect values of only 2 fields from the BING result:

1. The number of matches of the query word in the database (absolute document frequency).
2. The urls of the top 4 results retrieved for that query word.

Coverage and Specificity is calculated for every category and the database is classified as a particular category if it has coverage and specificity greater than both the threshold values.

Now we have a bunch of urls and we know the classification of the database. Using this we have formed a set of unique urls and crawl through each url using lynx to obtain the a set of documents which form the representative sample document set.

Before caching the documents we have filtered the contents of the documents to eliminate irrelevant content as well as invalid content. Also we have used SHA hashing to create a unique file name for every document making retrieval much more convenient and faster.

A main dictionary is formed that contains all unique words obtained from all documents of the representative sample document set. For every word in the main dictionary, we now find out the number of documents from the representative sample document set in which the word appears, i.e. the document frequency of the word in the representative sample document set.

The content summary consists of the words (in alphabetical order) their document frequency in the representative sample document set and their absolute document frequency. Since absolute document frequency is known only for all words that were also query words previously, -1 is written for words that were not query words.

Note: In the list of query words there were few phrases as well, we have **not** included them in the content summary.

1. **Result and Output Files**

The result of classification is shown in the terminal itself. For content summary text files are created at the end of the program execution. For e.g. the content summary for the database “health.com” at the Root level will be named as “Root-health.com.txt” and stored in the project directory itself. If the database has also been classified under Health then another file named “Health-health.com.txt” will also be created. A cache folder is also created that stores all the documents that have been crawled through.

1. **Your Bing Account Key -**  2dyKIv94jDETd7ClbVKoHvJSWFJ73ZvZRc7rjpBdkG8