HANDLING DUPLICATE DATA DETECTION OF QUERY RESULT FROM MULTIPLE WEB DATABASES USING UNSUPERVISED DUPLICATE DETECTION WITH BLOCKING ALGORITHM

Ms. Girija. M

*[*1]Assistant Professor, Department of Computer Science Auxilium College (Autonomous), Vellore, Tamil Nadu, India*

---***---

Abstract - ***The results from multiple databases compose the deep or hidden Web, which is estimated to contain a much larger amount of high quality, usually structured information and to have a faster growth rate than the static Web. The system that helps users integrate and more importantly, compare the query results returned from multiple Web databases, an important task is to match the different sources' records that refer to the same real-world entity. After removal of the same source duplicates, the assumed non duplicate records from the same source can be used as training examples. Unsupervised Duplicate Detection (UDD) uses two cooperating classifiers, a Weighted Component Similarity Summing (WCSS) classifier and a Support Vector Machine (SVM) classifier that iteratively identifies duplicates in the query results from multiple Web databases. For String Similarity calculation UDD uses any kind of similarity calculation method. Various experiments are conducted on a dataset to verify the effectiveness of the unsupervised algorithm in general and the additional blocking classifier in particular. Blocking is a technique to group data. Normally in order to classify records in a table, a unique hash function is generated for each record and compared with all other records in the table. Records having the same or similar hash value are categorized into groups. These groups are categorized as duplicates and non duplicates.***

Key Words: *Unsupervised Duplicate Detection, component Similarity summing, Support vector Machine*

I. INTRODUCTION

"A search engine is the window to the Internet". The basic premise of a search engine is to provide search results based on query from the user. A dynamic web page is displayed to show the results as well as other relevant advertisements that seem relevant to the query. This forms the basic monetization technique used by many popular search engines. The search engine contains a database that stores these links to the web pages and a framework to decide the sequence/order these results are displayed. With the exponential growth of the web pages and end users demand for optimal search results, there has been a huge push in using data mining techniques to perfect the process of understanding the data as well as pre-processing and data preparation. When dealing with large amount of data, it's important that there be a well defined and tested mechanism to filter out duplicate results. This keeps the end results relevant to the queries.

Duplicate records exist in the query results of many Web databases, especially when the duplicates are defined based on only some of the fields in a record. Using exact matching technique as part of preprocessing, records that are exactly the same in all relevant matching fields can be merged. The techniques that deal with duplicate detection can be broadly classified as those requiring training data (supervised learning method) and those that can function without a predefined training data (un-supervised learning method). As part of thesis plan to explore unsupervised techniques and develop or propose a mechanism that can function with minimal supervision. The premise of using web search engine example is to highlight the need for an algorithm that can handle large amounts of data and be able to derive a unique set that is most relevant to the user query.

Problem Statement

The end user has no control over the results returned by a search engine, nor can be guarantee that there will be no duplicates from the query result. The problem of duplicate records existing in a query result referring to the same real-world entity can occur when search engine uses multiple web databases. In the thesis focus on Web databases from the same domain, i.e., Web databases that provides the same type of records in response to user queries. Suppose there are *s* records in data source *A* and there are *t* records in data source *B*, with each record having a set of fields/attributes. Each of the *t* records in data source *B* can potentially be a duplicate of each of the *s* records in data source *A*. The goal of duplicate detection is to determine the matching status, i.e., duplicate or non-duplicate, of these *s* * *t* record pairs.

Possible Applications

One of the possible applications of duplicate detection would be data mining whose aim is to collect data from various sources and present the same in easy to understand reports to end users. The challenge of such systems is to be able to identify duplicate data that is being processed and not including the same in the output. The other major area and very relevant is the Master Data Management (MDM) applications. These applications are to serve as a central repository of master data in an organization and the main task is to filter duplicate records that refer to the same real-world entity.

MDM is used to manage non-transactional data in an organization for example customer or material master information. When there is a need to create for example a new customer, MDM system verifies if an account already exists with the given attributes and only then allow for new customer to be created. This process streamlines information and will avoid multiple accounts being created which refer to the same entity.

II. RELATED WORK

Record Matching Technique

Primarily record matching techniques can be broadly classified into the following

- Character or string based
- Token based
- Phonetic based
- Numeric similarity

1 Character or String Based Similarity Metrics

These set of techniques deal with various ways in comparing strings and finding a similarity metric that can be used to group as well as identify potential duplicates. There are many papers published and algorithms that were developed in analyzing the correct technique for comparing strings and arriving at a differentiating factor in order to measure their similarity.

The character-based similarity metrics are designed to handle common typographical errors. A typographical error (often shortened to typo) is a mistake made in the typing process (such as spelling) of printed material. Historically, this referred to mistakes in manual type-setting (typography). The term includes errors due to mechanical failure or slips of the hand or finger, but excludes errors of ignorance, such as spelling errors. String based similarity metrics measure how similar (or dissimilar) two strings are, two strings are deemed identical if they have same characters in the same order.
The following are the commonly used string based similarity metrics:

- Edit distance
- Smith-Waterman distance
- Jaro-Winkler distance metric

2 Token Based Similarity Metrics

Character based comparison work effectively in catching typographical errors, but they sometime fall short when comparing a rearranged string that has the same meaning. For example when comparing “Jane Doe” to “Doe, Jane”, characters based metrics fail and wrongly classify the two strings being different even though they refer to the same person name.

In order to avoid such error token based similarity measures are used, where comparison of two strings is done first by dividing them into a set of tokens (a token is a single word). A common practice is to split the string at white space and form the tokens. Thus in our example the string “Jane Doe” becomes a token array [“Jane”, “doe”]. Below are some the popular token based similarity metrics:

- Jaccard coefficient
- Cosine Similarity
- q-GRAMS

3 Phonetic Similarity Metrics

Strings may be phonetically similar even if they are not similar at character or token level. For example the word “*Color*” is phonetically similar to “*Colour*” despite the fact that the string representations are very different. The phonetic similarity metrics try to address such issues.

- Soundex
- Metophone

4 Numeric Similarity Metrics

There are numerous approaches that have been developed for comparing strings. When comparing numerical values the methods are primitive. In most cases when it makes sense to compare numbers, it’s a basic comparison and queries can be developed to extract numerical data with ease. There has been continuing research in using cosine similarity and other algorithms in analyzing numerical data.

For example data in numbers can be compared with primitive operators like equal, greater than and can used to calculate the difference between two numeric strings.

III. NEW PROPOSED UDD MODEL

This Unsupervised Duplicate Detection (UDD) with blocking system consists of four modules: data retrieval, preprocessing and blocking, the main algorithm and data presentation. The same can be seen in fig 3.1 below is a simple architecture diagram illustrating the system.

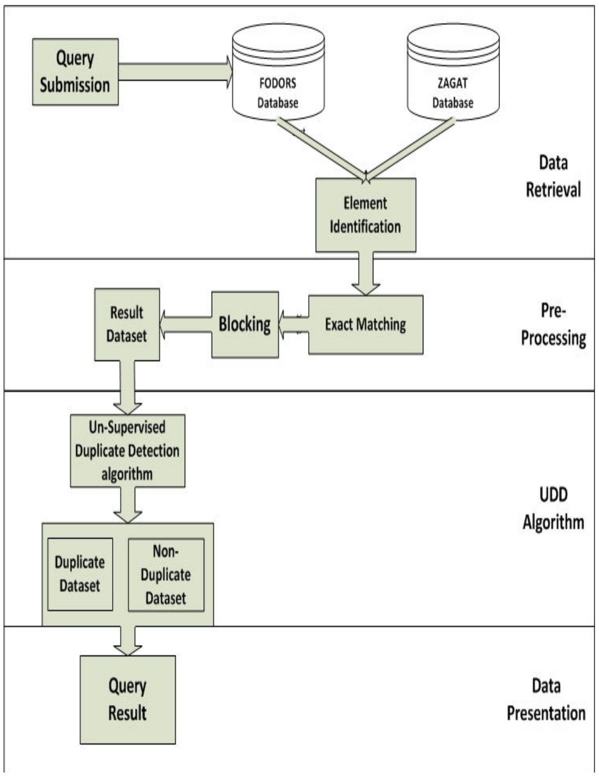


Fig: 1.1 Unsupervised Duplicate Detection (UDD)

A. Data Retrieval

The data retrieval module consists of an interface to read the user query along with the actual data retrieval from the database. To make the application interactive and simulate a real world application a search box is given where the user can enter a query which can be for a particular word or run wide open to query all the records in the database for analysis.

The data exists in two tables and has the same structure (field names). Element identification is the process of mapping the fields of two tables and reading the information. Dealing with a simple dataset (explained in section 3.2) with fixed field names. This may not be the case in a real world application where data can be stored in databases with different names/elements and the process of mapping fields between the data sources is a very crucial step in ensuring the integrity of the application.

B. Pre-Processing and Blocking

The second module in the application consists of pre-processing and blocking. In this step generally data is cleansed and parsed into one structure. This step also involves removing special characters from the raw data and converting them to a lower case for accurate comparison. As part of pre-processing, data is sorted and exact matching records are deleted. This is done comparing all the fields (taking each row as one string) and comparing it to others. This ensures the same data doesn't exist and is a basic check that can be done.

The next major and crucial step in this module is blocking. Blocking "typically refers to the procedure of subdividing data into a set of mutually exclusive subsets (blocks) under the assumption that no matches occur across different blocks". Normally in order to classify records in a table, a unique hash function is generated for each record and compared with all other records in the table. Records having the same or similar hash value are categorized into groups. The details and techniques that are used for blocking and also look at the efficacy of each of them.

C. **UDD Algorithm**

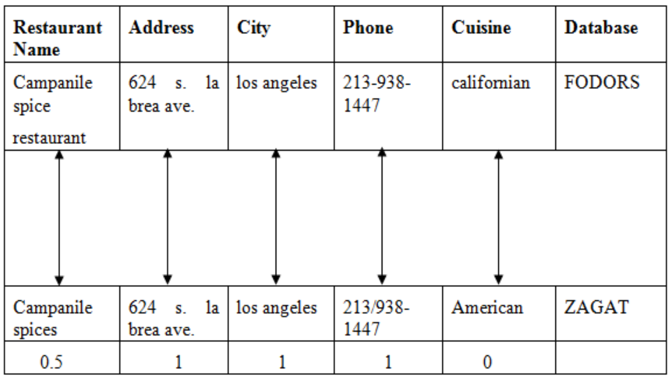
The main component of the system is the module that has the UDD algorithm. In this step look at developing an algorithm that can train itself and aid in identifying duplicates. This algorithm consists of a component that calculates the similarity vectors of the selected dataset, assigning weights to the selected vectors and finally using the Support Vector Machine (SVM) to classify the data. The technical details and further information on each of these components is discussed at length later in this section.

D. Data Presentation

The final module consists of presenting the data to the user. The unique data along with statistics is presented to the user.

Similarity Vector Calculation

The next step is the similarity vector calculation which holds comparison of two records. Inputs to this process are the potential duplicate dataset and non-duplicate dataset (outputs of the blocking classifier). The output of a similarity vector function is a set of attribute similarity scores for each pair of records in the dataset. In this step, the UDD algorithm calculates the similarity of record pairs in both datasets grouped by the blocking classifier. The output of this process serves as input to Weighted Component Similarity Summing (WCSS) Classifier and SVM classifier which are examined in detail in the following sections. For example let's consider two records, one from ZAGAT and another from FODORS.



Restaurant Name	Address	City	Phone	Cuisine	Database
Campanile spice restaurant	624 s. la brea ave.	los angeles	213-938-1447	californian	FODORS
↕	↕	↕	↕	↕	
Campanile spices	624 s. la brea ave.	los angeles	213/938-1447	American	ZAGAT
0.5	1	1	1	0	

Fig. 3.3 Example of Similarity vector calculation

As can be seen from the above example, for a pair of records each of the fields/attributes are compared individually (e.g. Restaurant Name with Restaurant Name, City with City etc.). By doing this easily to get better establish two record's similarity or lack of it. Using the same above example (Fig. 3.3 Example of Similarity vector calculation) already knows the records refer to the same real world entity and the only differentiating attribute is the cuisine value. The similarity vector value for this pair of records would be <0.5,1,1, 1,0>. Note that even though the value for phone attribute is stored differently, this was resolved by ignoring text differences (In this thesis used a text formatting routine to remove special characters). To calculate the similarity scores between the two field/attribute values first to determine the number of transformations that can be performed between two strings.

IV. SYSTEM IMPLEMETNATION

Attribute Weight Assignment

In the WCSS classifier, assign weight to an attribute to indicate its importance. The weights of a field/attribute are given in such a way that the sum of all fields/attributes weights is equal to 1. In non-duplicate vector most of the fields will have small similarity score (Table 3.8) for all record pairs where as in duplicate vector most of the fields will have large similarity score (values will be similar to Table 3.7) for all record pairs. In general, WCSS classifier employs duplicate and non-duplicate intuitions for assigning weights. Inputs to this function are the similarity vector of non-duplicate records and duplicate records.

Duplicate Intuition

For duplicate records the similarity between them should be close to 1. For a duplicate vector V12 that is formed by a pair of duplicate records r1 and r2, assign large weights to the fields with large similarity values and small weights to the fields with small similarity values. This will ensure that a record pair with most similarity gets classified as duplicates.

$$w_{di} = \frac{p_i}{\sum_{j=1}^{m} p_j} \quad \text{// m is the number of fields in dataset}$$

$$p_i = \sum_{k=1}^{n} v_{ki} \quad \text{// n is the number of records in duplicate vector set}$$

Equation 1: Weight calculation for all the fields using duplicate vectors.

Where,

W_{di} = Normalized weight for ith attribute.
p_i = Accumulated i^{th} attribute similarity value for all duplicate vectors.

For each attribute, pi value will be large when it has a large similarity value in the duplicate vector, which will result in a large weight value being assigned to ith field. On the other hand, the field will be assigned a small weight if it usually has a small similarity value in the duplicate vectors

Non-Duplicate Intuition

In non-duplicate records the similarity between them should be close to 0. Hence, for a non-duplicate vector V12 that is formed by a pair of non-duplicate records r1 and r2, we need to assign small weights to the fields with large similarity values and large weights to the fields with small similarity values. This will ensure that a record pair with less similarity gets classified as non-duplicates.

$$w_{ni} = \frac{q_i}{\sum_{j=1}^{m} q_j} \quad \text{// m is the number of fields in dataset}$$

$$q_i = \sum_{k=1}^{n} (1 - v_{ki}) \quad \text{// n is the number of records in non-duplicate vector set}$$

Equation 2: Weight calculation for all the fields using duplicate vectors.

Where,

w_{ni} = Normalized weight for ith attribute
qi = Accumulated ith attribute dissimilarity value for all non-duplicate vectors

In non-duplicate vectors the dissimilarity value of ith field is 1-vi (where vi is the similarity of ith field). For each field, if it usually has a large similarity value in the non-duplicate vectors, it will have a small accumulated dissimilarity (qi) and will, in turn, be assigned a small weight. On the other hand, it will be assigned a large weight if it usually has a small similarity value in the non-duplicate vectors.

The final weight of attribute is the combination of two intuitions weighting schemes. As part of experiments, each scheme was given a weight to show its importance:

$$w_i = a * w_{di} + (1 - a)w_{ni}$$

Equation 3: Weight of all the fields combining duplicate weight and non duplicate weight.

Duplicate Identification

Once to get the weights of each field and the similarity vectors of non-duplicate and potential duplicate datasets, the duplicate detection can be done by calculating the similarity between the records. Hence, define the similarity between records as:

$$Similarity(r_1, r_2) = \sum_{i=1}^{n} w_i * v_i$$

Equation 3.5: WCSS similarity for records r1 and r2

- Where r1, r2 are the two records for which the similarity is being calculated.
- w_i is the weight of field(i).
- v_i is the similarity vector of two records r1, r2 of field(i) .

1. Support Vector Machine (SVM) Classifier

Seen earlier in chapter 2, SVM classifier is a tool used to classify data. SVM uses a two-step process, training and classification. In the training step, labeled data is supplied to the classifier, labeling each record as either positive or negative. SVM internally plots the information (fig. 2.3) by separating the data into groups. During the classification step, when the system is supplied with data to be classified, it classifies the record as either being positive or negative based on the training data.

The WCSS classifier outputs three sets of similarity vectors namely potential duplicate vectors, non-duplicate vectors and identified duplicate vectors. From these vectors, the identified duplicate vectors D are sent as positive examples and non-duplicate vectors N as negative examples for training purpose. These two vectors serve as input (training data) to the SVM classifier. This can be train SVM classifier and use this trained classifier to identify new duplicate vectors from the potential duplicate vector P.

Training / Learning

Training data for SVM classifier is the similarity vectors from duplicates (identified by the WCSS classifier) and non-duplicates. Duplicates are labeled as positive and non-duplicates as negative.

+1 1:1.000	2:0.8000	3:1.000	4:1.000	5:0.000
+1 1:0.5	2:1.000	3:1.000	4:1.000	5:0.000
-1 1:0.000	2:0.000	3:1.000	4:0.000	5:0.000
-1 1:0.000	2:0.000	3:1.000	4:0.000	5:0.000
-1 1:0.000	2:0.000	3:1.000	4:0.000	5:0.000
-1 1:0.000	2:0.000	3:1.000	4:0.000	5:0.000

Table 3.9: Training data for SVM classifier combing entries from table 3.5 and table 3.6 Classification

Once training is complete, similarity vectors from potential duplicate dataset are sent to the SVM classifiers classify function to determine if they are duplicates. The output from the classify function is either a positive value (duplicate) or negative value (non-duplicate).

Next chapter contains details about various experiments that were conducted and how SVM classifier was able to classify data.

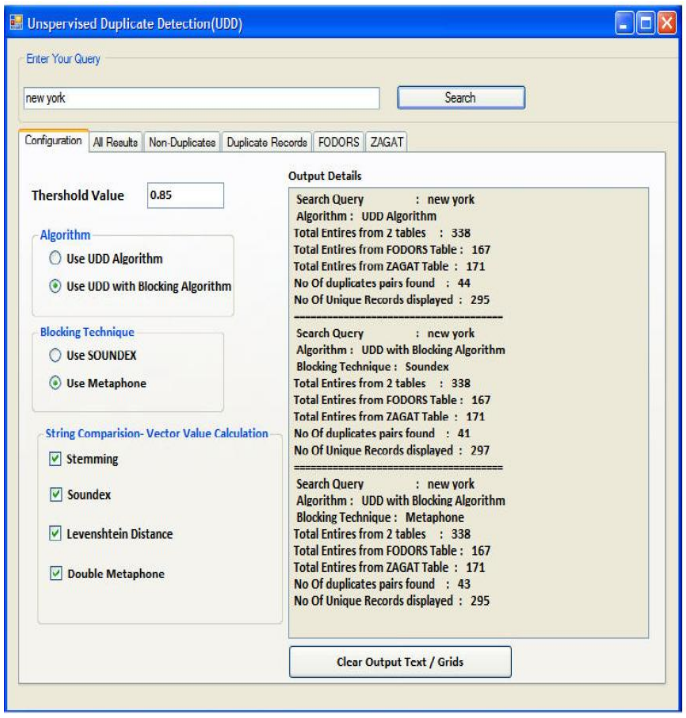
V. EVALUATION RESULT:

To analyze the effectiveness of the UDD algorithm in identifying duplicates various experiments were conducted using the restaurant dataset. From the original dataset already seen that there are 112 identified duplicates pairs. The application that was developed for this thesis includes various configuration options (as described in section 4.2) that need to be tested and results validated. Experiments were divided into two categories, one involved querying using random words (experiments 1-6) and other is the analysis of the complete restaurant dataset.

1 Random Query Experiments

Experiment 1

A search query = "new York" using the standard UDD algorithm and UDD algorithm with blocking techniques (Soundex and Metaphone). By default chose all the string comparison functions (Stemming, Soundex etc) and let the threshold value be at 0.85.



new york

Thershold Value 0.85

Algorithm
- ( ) Use UDD Algorithm
- [✓ Use UDD with Blocking Algorithm]

Blocking Technique
- ( ) Use SOUNDEX
- [✓ Use Metaphone]

String Comparision- Vector Value Calculation
- ☑ Stemming
- ☑ Soundex
- ☑ Levenshtein Distance
- ☑ Double Metaphone

Output Details

```
 Search Query             :  new york
 Algorithm :  UDD Algorithm
Total Entires from 2 tables    :  338
Total Entires from FODORS Table :  167
Total Entires from ZAGAT Table  :  171
No Of duplicates pairs found    :  44
No Of Unique Records displayed  :  295
-----------------------------------------
 Search Query             :  new york
 Algorithm :  UDD with Blocking Algorithm
 Blocking Technique :  Soundex
Total Entires from 2 tables    :  338
Total Entires from FODORS Table :  167
Total Entires from ZAGAT Table  :  171
No Of duplicates pairs found    :  41
No Of Unique Records displayed  :  297
=========================================
 Search Query             :  new york
 Algorithm :  UDD with Blocking Algorithm
 Blocking Technique :  Metaphone
Total Entires from 2 tables    :  338
Total Entires from FODORS Table :  167
Total Entires from ZAGAT Table  :  171
No Of duplicates pairs found    :  43
No Of Unique Records displayed  :  295
```

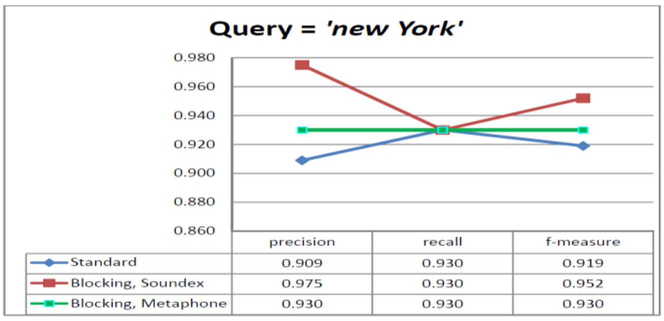
Fig. Experiment 1 for search query 'new york' output details of all algorithms

As can be seen from the above Fig 5.1, now derive from the following evaluation metrics.

	Number of duplicate pairs found	Correctly identified duplicate pairs	True duplicate pairs	Precision	Recall	f-measure
UDD Algorithm	44	40	43	0.909	0.930	0.919
UDD with Soundex	41	40	43	0.975	0.930	0.952
UDD with Metaphone	43	40	43	0.930	0.930	0.930

Table: Precision, Recall and f-measure for search query 'newyork'

Above table or the below graph precision, recall and f-measure are better in blocking algorithm using Soundex than the standard UDD algorithm. It is also observed that blocking with Metaphone works better than the standard UDD algorithm.

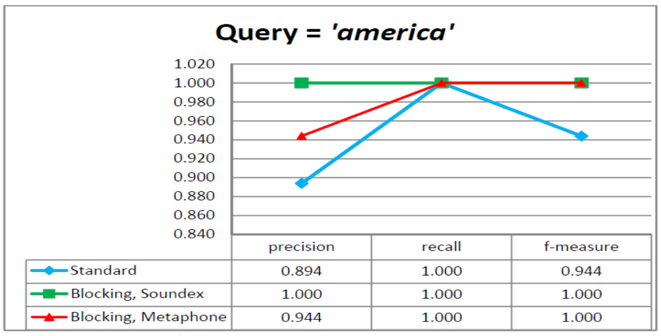


	precision	recall	f-measure
Standard	0.909	0.930	0.919
Blocking, Soundex	0.975	0.930	0.952
Blocking, Metaphone	0.930	0.930	0.930

Fig. 2 Graph showing the evaluation metric details for search query 'new york'

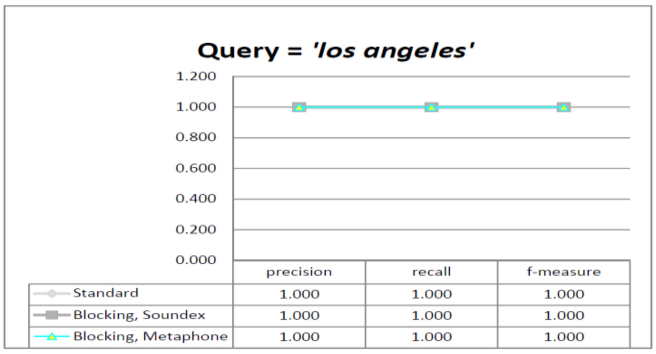
Now let's consider search query = "america" using the standard UDD algorithm and UDD algorithm with blocking techniques (Soundex and Metaphone). By default chose all the string comparison functions (Stemming, Soundex etc) and let the threshold value remain at 0.85.

For this experiment, all the metrics -precision, recall and f-measure are better in the UDD with blocking algorithm as compared to the standard UDD algorithm



	precision	recall	f-measure
Standard	0.894	1.000	0.944
Blocking, Soundex	1.000	1.000	1.000
Blocking, Metaphone	0.944	1.000	1.000

Fig: Graph showing the evaluation metric details for search query 'America'



	precision	recall	f-measure
Standard	1.000	1.000	1.000
Blocking, Soundex	1.000	1.000	1.000
Blocking, Metaphone	1.000	1.000	1.000

Fig: Graph showing the evaluation metric details for search query 'los angeles'

For this experiment, both the algorithms, UDD and UDD with blocking have the same metrics - precision, recall and f-measure. The standard UDD algorithm found a total 139 duplicate pairs with 41 false positives (wrongly classified by the classifier as duplicates). After accounting for this we have an accuracy of 87.5% and f-measure of 0.780.

UDD with blocking classifier produced similar results when using Soundex and Metaphone techniques. When compared to the standard UDD algorithm, the blocking algorithm identified 122 duplicate pairs with 21 false positives. The accuracy rate is 90.2% which is better than the standard UDD algorithm. The main difference is the reduction of false positives. Blocking algorithm classified 21 false positives when compared to 41 by the standard UDD. This is a big improvement and helps in reducing wrong classification of records as duplicates**.**

CONCLUSION

This Paper concentrated on the development of an Unsupervised Duplicate Detection algorithm that can serve as foundation for developing applications that use Web databases. Seen from the results, using an additional classifier (like blocking) can result in higher accuracy. With exponential growth of data, duplicate detection is an important problem that needs more attention, using an UDD algorithm that learns to identify duplicate records has some advantages over offline/supervised learning methods. Although the focus of the UDD application in the thesis was limited to restaurant dataset, the same principles can be used broadly to other domains. When compared to traditional databases, Web-based retrieval system in which records to match are greatly query-dependent, a pre-trained approach is not appropriate as the set of records in response to a query is a biased subset of the full data set. UDD algorithm which is an unsupervised, online approach for detecting duplicates is a suitable solution, when query results are fetched from multiple Web databases. The core of UDD algorithm relies on using WCSS and SVM classifiers to assign weights and classify data. This thesis is a step forward in enhancing the UDD algorithm by adding an additional classifier.

Similarity metrics forms the basis for determining the similarity of two strings/objects. There is a need to develop new algorithms that can present accurate results while take escaping into account various forms of data. These days there are many applications that extract and present information from the Web. This information is either unstructured or imprecise. Duplicate record detection techniques are crucial for improving the quality of the extracted data. This calls for development of robust and scalable solutions. More research is needed in the area of data cleaning and information quality in general and in the area of duplicate record detection in particular.

REFERENCES:

1. Fayyad, Usama; Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996) "From Data Mining to Knowledge Discovery in Databases".
2. SB Kotsiantis, “Supervised learning: A review of classification techniques” Informatica, vol. 31, pp. 249–268, 2007.
3. R.Baxter, P. Christen, and T. Churches, “A Comparison of Fast Blocking Method for Record Linkage”, Proc. KDD Workshop Data Cleaning, Record Linkage and Object Consolidation, pp. 25-27, 2003.
4. W. E. Winkler. The state of record linkage and current research problems. Technical Report RR99/04, US Census Bureau, 1999.
5. I.P Fellegi and A. B. Sunter. A theory for record linkage. Journal of the American Statistical Association, 40, 1969.
6. M. A. Hernandez ´ and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 2(1):9–37, 1998.
7. V.S. Verykios, G.V. Moustakides, and M.G. Elfeky, “A Bayesian Decision Model for Cost Optimal Record Matching,” The VLDB J., vol. 12, no. 1, pp. 28- 40, 2003.
8. A. McCallum, K. Nigam, and L. H. Ungar. “Efficient clustering of high-dimensional data sets with application to reference matching”. In KDD, pages 169–178, 2000.
9. R. Ananthakrishna, S. Chaudhuri, and V. Ganti. “Eliminating fuzzy duplicates in data warehouses”. In VLDB, pages 586– 597, 2002.
10. S. Chaudhuri, V. Ganti, and R. Motwani, “Robust Identification of Fuzzy Duplicates” , Proc.21st IEEE Int’l Conf. Data Eng., pp. 865876,2005.