A Holistic Schema Matcher Application

GRADUATE PROJECT REPORT

Submitted to the Faculty of
The Department of Computing Sciences
Texas A & M University- Corpus Christi
Corpus Christi, Texas

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
By

Archana Gudala
Fall 2010

Committee Members

Dr. Longzhuang Li
Committee Chairperson

Dr. Mario Garcia
Committee Member

Dr. Hongyu Guo
Committee Member
ABSTRACT

The capability of matching the schemas of the databases has led to its wide use in e-commerce, data integration, data warehousing, query processing etc. Thus, due to its increasing demand, a large subsequent set of research is happening in this field. In order to support this research we have developed an application, A Holistic Schema Matcher, which could provide a match between two or three schemas. It implements the basic concepts from element-level techniques as well as instance-level techniques to provide the results for the match. This is a user-friendly, standalone desktop application which matches the attributes of the schema semantically related to each other.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>2</td>
</tr>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>6</td>
</tr>
<tr>
<td>1. Background and Rationale</td>
<td>7</td>
</tr>
<tr>
<td>1.1 Applications of Schema Matching</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Existing Methods of Schema Matching</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Comparison of Schema Matching Algorithms</td>
<td>15</td>
</tr>
<tr>
<td>2. Narrative</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Approaches of Schema Matching</td>
<td>19</td>
</tr>
<tr>
<td>2.1.1 Schema-level Matchers</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Instance-level Matchers</td>
<td>22</td>
</tr>
<tr>
<td>2.1.3 Hybrid Matchers</td>
<td>23</td>
</tr>
<tr>
<td>2.1.4 Composite Matchers</td>
<td>24</td>
</tr>
<tr>
<td>2.2 Types of Schema for Matching</td>
<td>24</td>
</tr>
<tr>
<td>2.3 Project Data</td>
<td>26</td>
</tr>
<tr>
<td>2.4 Tool developed as part of the project</td>
<td>29</td>
</tr>
<tr>
<td>2.5 Scope of the project</td>
<td>29</td>
</tr>
<tr>
<td>2.6 Java Swings Jargon</td>
<td>30</td>
</tr>
<tr>
<td>3. System Design</td>
<td>32</td>
</tr>
<tr>
<td>3.1 System Requirements</td>
<td>33</td>
</tr>
<tr>
<td>3.2 System Architecture</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Flowchart of Holistic Schema Matcher</td>
<td>42</td>
</tr>
</tbody>
</table>
3.4 Demo of the Tool.................................................................43

4. Testing and Evaluation......................................................50
   4.1 Testing........................................................................50
   4.2 Results Analysis............................................................54

5. Future Enhancements.........................................................58

6. Conclusion........................................................................60

Bibliography and References....................................................61

A. Appendix – A.................................................................63

B. Appendix – B.................................................................65

C. Appendix – C.................................................................67
LIST OF FIGURES

Figure 1.1: Representation of Schema in COMA..............................................11
Figure 1.2: Representation of XML Schemas in CUPID.................................12
Figure 1.3: A sample rooted graph structure for a real-world scenario.............13
Figure 1.4: Example of Similarity Flooding Algorithm....................................14
Figure 2.1: A Simple Example of Schema Matching.......................................19
Figure 2.2: Classification of Schema Matching approaches.............................20
Figure 2.3: Sample Schemas for Purchase Order...........................................24
Figure 2.4: The observing stations of TCOON at the coastal regions...............27
Figure 2.5: Schematic diagrams of the TABS buoy models............................28
Figure 3.1: System Architecture.................................................................33
Figure 3.2: Results generated based on the formulae.....................................39
Figure 3.3: Flowchart depicting the execution of the application ....................43
Figure 3.4: The GUI of Holistic Schema Matcher Application..........................44
Figure 3.5: The output generated from element-level matching techniques.......45
Figure 3.6: The output generated from instance-level matching techniques......46
Figure 3.7: The output generated from holistic schema matching....................47
Figure 3.8: The output generated by schema 2 and schema 3..........................48
Figure 3.9: The output generated by schema 3 and schema 1..........................49
Figure 4.1: Results obtained from Test Case 1.............................................51
Figure 4.2: Results obtained from Test Case 2.............................................52
Figure 4.3: Results obtained from Test Case 3.............................................53
LIST OF TABLES

Table 1.1 Summary of the evaluations...............................................................16
Table 4.1 Results obtained from element level matching techniques.................54
Table 4.2 Results obtained from instance level matching techniques...............54
Table 4.3 Results obtained from Holistic level matching techniques...............55
Table 4.4: Results obtained from element level matching technique...............55
Table 4.5: Statistics from the output of Instance level matching......................56
Table 4.6: Results obtained from Holistic matching techniques.......................56
Table 4.7: Results obtained from element level matching for Movies domain.......57
Table 4.8: Results obtained from instance level matching for Movies domain.....57
Table 4.9: Results obtained from holistic matching for Movies domain.............57
1. BACKGROUND AND RATIONALE

Schema matching is an approach which was in research since late 1970s. Many researchers and scientists have focused on schema matching as its primary research interest in order to find a fully automatic schema matching system which is domain independent. Indeed, it didn’t succeed till now due to various factors such as heterogeneities of schema and structures. Schema matching is considered important in almost all the fields like data warehousing, data integration, e-commerce and semantic query processing which interacts with other schemas of several different systems. In today’s world, an individual cannot exist without a computer, they have to always depend on the internet for getting any information available in large junks which is incomparable with the other systems. Hence, the basic need of schema matching comes into picture. A match among the various systems needs to be performed in order for the end-user to easily co-relate the data from one system to another system and make his decisions.

These schema matching systems cannot judge on its own without the help of a human intervention. Hence, semi-automatic schema matching techniques are currently being used nowadays which needs lesser percentage of human interaction. Many semi-automatic schema matching systems primarily focus on only one techniques rather than leveraging several individual matching techniques. This project has been proposed to implement a holistic schema matching which performs both instance level and element level based matching techniques on the Coastal studies data like TCOON, TABS etc.

The Gulf of Coastal studies are making enough arrangements in installing several measurements systems across several locations in the coastal regions to measure the real-
time water level and other related meteorological information which really proves helpful for the Corpus Christi officials in order to identify the tropical storms and hurricanes ahead and take effective measures in order to save the human lives. Although the data would be available to the officials from several regions, the schema would be different from one region to another. Hence, a system is implemented to match the elements as well as the data from these regions so that it would be helpful for them. Otherwise, it’s a time consuming process to match each of the schemas to get the required information.

To facilitate the process to gather data from multiple ocean observing systems, we have developed a system to match two or three schemas at the same time, provided the data is given both in text and excel. We are planning to extend our project to more than three schemas and also relate the data from different files as part of our future enhancements.

1.1 Applications of Schema Matching

This section explains the various database applications which are in immediate need of schema matching. Schema matching is of ultimate help for these database applications [Do 2005]. Some of them are discussed briefly below.

1.1.1 Data Integration:

This can be considered as the most important motivation for schema matching. The problem of schema matching has been into existence since mid 70s. Data integration needs to integrate the data from several different sources which tend to have different schematic structure and it is a huge difficulty to find a match between the sources. Hence, schema matching came into picture in this context. Since the schemas, ontologies and the XML formats are developed by different individuals who don’t coincide with each other,
there is no possibility to find a match between them. Thus, schema matching will convert
the local schemas i.e. the schema structure of each of the sources into a generic format
accessible to everyone called global schema. Apart from matching the schemas of each of
the sources, schema matching is also involved in matching the instance data in these
schemas. Semi-automatic methods and approaches are currently in place which still needs
human effort.

1.1.2 E-business:

The rapid growth of internet into business urges for the immediate implementation of
schema matching into their domain. Due to the enormous amount of data that is available
online, humans tend to use the existing data on the web which could not coincide with
similar information. Hence, schema matching acts as an intermediate in matching the
business formats and making it accessible to everyone. These businesses usually handle
various types of transactions, product information which is not similar with other
business in the same domain. This information has to be exchanged from different
business people which leads to the concept of schema matching. Hence, all the message-
formats need to be made generic which is the current bottleneck. Semi automatic schema
matching methods tend to reduce the amount of overhead by generating the
transformation rules between the two schemas. Thus, schema matching has its
importance in this field as well.

1.1.3 Data warehouse:

Data warehouse integrates the data from several heterogeneous data sources. These
data sources are developed by individuals from different organizations which follow
different standards. Data warehouse is the key practice of maintaining the data from
organizations across the world and generate reports to analyze and spread their business. Hence, when the actual part of maintaining the data warehouse comes into picture, there will be all sorts of problems in finding a match among the schemas. This is where schema matching will play its vital role. The schema matching will match the data from several sources and then it wouldn’t be of enough trouble in maintaining a data warehouse.

1.1.4 Sequential Query Processing

This application is quite different from data integration, data warehouses and E-commerce. These applications use schema matching during their design phase which is different from the semantic query processing which uses the process during their run-time scenario. In this scenario, the user gives the required output of the query, and the system identifies a process to match the output. Hence, in this case they use schema matching. The user’s specifications and the attribute names given to these schemas might be different even though the content is the same; hence they use the match operation to find a match between the schemas. Hence, schema matching is thus used in this scenario at run-time analysis [Wikipedia 2010].

1.2 Existing Methods of Schema Matching

There are several methods which are currently being used for schema matching. Few of those methods are explained as follows.

1.2.1 COMA (Combining Matcher):

COMA is a generic match system which supports different applications and a variety of schemas and XML which combines all the matchers in a flexible way. There are a lot of efforts that have been made by the researchers in developing a fully automatic schema matching which had to combine several matching algorithms. The
main issue arises while trying to combine the various matching algorithms [Do 2005]. Thus, COMA will act as an intermediate to perform this schema matching by using all the available matching algorithms in the library. Schemas are represented as directed acyclic graph and the attributes are represented as graph nodes connected by directed links. The schemas are imported and then a match algorithm will be performed on the inputs available. [Rahm 2002] This system has a library which contains a huge list of all the matching algorithms and there is always an option to add new algorithms in the library so that new matching’s can also be performed. The representation of a relational schema in terms of acyclic graph is represented in Figure 1.1.

![Figure 1.1: Representation of Schema in COMA](image)

It serves as a platform to verify and compare the effectiveness of these matches and combine various match results. COMA is not a fully automatic schema matching approach because of its inadequacy in solving the heterogeneity between the schemas. Hence, manual interpretation plays an active role in judging the proposed matches by the system. The main advantage of COMA is its reusability. The concept of reusing the
available solution will reduce the amount of overhead in terms of time and it is less prone to errors since it is already in place.

### 1.2.2 CUPID

CUPID is a hybrid matcher combining several matching methods. It has been applied to XML and relational schemas [Philip 2001]. It can be represented using a tree-representation and it is also extended to use rooted graphs. CUPID maps the schema elements based on their names, data types, constraints, etc by using linguistic matching and structural matching. It mainly focuses on the schema based matching rather than the instance based matching [Jayant 2001]. It makes use of some of the characteristics of the past algorithms like ARTEMIS [Artemis 2010], SEMINT [Wen 1999], etc. in their matching methods.

![Figure 1.2: Representation of XML schemas in CUPID.](image)

Matching in CUPID is carried out in three major steps, i.e. Firstly, the element names are broken down into tokens and are expanded to identify their abbreviations based on a thesaurus, then they group the elements with similar data types and constraints and finds a similarity coefficient between these element which range from [0,1]. Secondly, the schema elements are transformed into trees and perform a structure matching. An example of the representation of the XML schemas is shown in Figure 1.2.
The assumption behind structure matching is that most of the vital information is present in the leaves and the leaves are less adapted to changes rather than the schemas [Jayant 2001]. The leaves are matched based on the structural similarity that is computed. Thirdly, a match is performed by the values that are generated in the second part.

Despite of its tree representation, the real-world schemas contain substructures and the referential constraints which are much better represented in rooted graphs. These rooted graphs have elements, nodes and relationships like aggregation, containment and references which is referred in Figure 1.3.

![Sample rooted graph structure](image)

Figure 1.3: A sample rooted graph structure for a real-world scenario.

The main drawback is that you have to do some background work on finding out the abbreviations in thesaurus. When a same set of XML is tested with COMA and CUPID, it showed better match correspondences than the existing methods [Hong-Hai 2002] like COMA, DIKE and MOMIS.

1.2.3 ARTEMIS (Analysis of Requirements; Tool Environment for Multiple Information Systems)

This system [Artemis 2010] is designed as an intermediate module of MOMIS mediator system in University of Milano. It clusters the schema attributes based on name
and structure affinities. This is a hybrid schema matcher which utilizes both the element-
level and structure level information of the input schemas. The name and structural
affinities are calculated based on inter-schema relationships or intra-schema relationships
using a common thesaurus [Philip 2001]. Global affinity coefficients is the measure of
the level of matching between the schema elements based on their names, characteristics
defined between them. Based on these coefficients, a clustering algorithm groups the
similar classes at varied levels of affinity, which in turn creates a cluster with a set of
global attributes called global class. Then, logical correspondences between the global
class and other schema attributes are determined using a mapping table.

1.2.4 Similarity Flooding (SF)

This schema matching [Wen 1999] converts the schemas into directed graphs and
uses fix point computation to analyze the correspondences between the nodes of the
graph. The algorithm just implements a simple name matcher which performs basic string
level matching techniques like prefix, suffix, postfix comparisons [Do 2005]. It does not
rely on any thesaurus like the other matching algorithms rather than just making use of
the string matching comparisons. It converts all the other types of schemas like SQL,
DDL, XSD, OWL into directed graphs and then performs the match criteria. It usually
takes two schemas as input and produces a mapping as an output which in turn will be
filtered based on the matching criteria and then a part of the mapping will be considered
as the final output. When the schemas are considered as directed graphs, each edge is
represented as (s, p, o) where s represents the source, o represents the target, and p
represents the label of the edge.

Consider an example as shown in Figure 1.4 with two schemas converted into directed graphs and termed as Model A and Model B. There are many steps involved for the computation the fixpoint [Sergey 2002]. As the initial step, a pairwise connectivity graph (PCG) will be computed from the auxiliary data that is derived from these tables, later an induced propagation graph will be developed which contains an opposite edge for each of the existing edge in the graph. Finally, a fixpoint computation will be computed and filters are implemented in order to find the final match of the two schemas A and B.

This method can be considered to be good based on the statistics that are computed [Hong-Hai 2002]. The Overall quality over all the match tasks is 0.6.

1.3 Comparison of Schema Matching Algorithms

Schema matching tools are developed for a particular scenario in real-world schemas and they yield better performance for those cases. Hence, it is not possible for anyone to judge the best among the tools that are developed for different domains [Fabein 2007]. Currently, there are many schema matching algorithms that are in use, but we are not in a position to classify which is the best due to the diverse ways of identifying its effectiveness of each system.
Table 1.1: Summary of the Evaluations

<table>
<thead>
<tr>
<th>References</th>
<th>Autoplex &amp; COMA</th>
<th>COMA</th>
<th>Cupid</th>
<th>LSD &amp; GLUE</th>
<th>Semilast</th>
<th>SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test problems</td>
<td>relational</td>
<td>XML</td>
<td>XML</td>
<td>relational</td>
<td>XML, relational</td>
<td></td>
</tr>
<tr>
<td>Match types</td>
<td>relational</td>
<td>XML</td>
<td>XML</td>
<td>relational</td>
<td>XML, relational</td>
<td></td>
</tr>
<tr>
<td>#Schemas / Match tasks</td>
<td>15/21 &amp; 15/22</td>
<td>5/10</td>
<td>2/1</td>
<td>24/20 &amp; 3/6</td>
<td>10/5</td>
<td>7/9</td>
</tr>
<tr>
<td>Min/Max/Avg schema size</td>
<td>40/145.77</td>
<td>40/54/47</td>
<td>14/66/ &amp; 24/133/145</td>
<td>6/260/57</td>
<td>5/22/12</td>
<td></td>
</tr>
<tr>
<td>Min/Max/Avg schema similarity</td>
<td>0.43/0.8/0.58</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.46/0.94/0.75</td>
<td></td>
</tr>
</tbody>
</table>

Match result representation
- element-level correspondences with similarity value in range [0,1]
- path

Element representation
- node
- attr

Local global cardinality
- 1/1:1
- 1/m-n

Quality measures and test methodology

Employed quality measure
- Precision, Recall, F-Measure
- Precision, Recall, Overall

Subj ectivity
- 1 user
- 7 users

Studied impact on match quality
- Automatch: methods for sampling instance data
- Matchers, combination, reuse, schema characteristics
- Specifying domain synonyms
- Specifying domain synonyms
- Specifying domain synonyms, constraints
- Learner combinations, LSD amount of data listings
- Constraints (discriminators)
- Filters, fix-point formulas, randomizing initial sim

Pre-match effort
- Training
- Specifying domain synonyms
- Specifying domain synonyms
- Specifying domain synonyms, constraints

Best average match quality

<table>
<thead>
<tr>
<th>Free, Recall</th>
<th>F-Measure</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.84/0.82</td>
<td>0.9/0.89</td>
<td>0.68</td>
</tr>
<tr>
<td>0.81 &amp; 0.72</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>0.66</td>
<td>0.43</td>
<td></td>
</tr>
</tbody>
</table>

Evaluation highlights
- Big schemas, systematic evaluation
- Comparative evaluation of 3 systems
- Big schemas, No pre-match effort
- Big schemas, No pre-match effort
- User subjectivity, No pre-match effort

Few of the existing approaches are explained in the previous section and would like to present the statistics of the matching algorithms to give us a clear understanding of where we really stand in our research for schema matching in Table 1.1. Various aspects related to the history of schema matching are discussed in this section. The basic concept that is underlying behind this application and the main implementation logic has been elaborated in the later sections.
2. NARRATIVE

Schema is the metadata of the database design. It contains the entire information of the attributes present in the database and instances that can be stored. Apart from the characteristics mentioned, it has some concerns relating to cardinality, integrity, referential constraints etc. Schemas may be developed using different schema definition languages. For Example, Structured Query Language (SQL) is for relational model, Document type definition (DTD) and XML Schema Definition (XSD) for XML schemas and Web Ontology Language (OWL) for Ontologies [Do 2005]. Each of these different types of schemas is developed for a particular application, but they all contribute to the use of schemas in their applications.

Schema matching is the process of finding semantic correspondences between the objects. It can be formulated as follows:” *Given two schemas \( S_1 \) and \( S_2 \), we have to find a plausible correspondence between elements of \( S_1 \) and \( S_2 \) exploiting all the information such as schemas, instance data*”. The main goal of the schema matching is to reduce the amount of manual effort and avoid human intervention in solving the matching problems from scratch. This process has gained profound interest in both research and practice. Matching the components either in database schemas or XML or Ontologies has become necessary for almost all the applications like peer to peer databases, data migration, data integration, and query processing.

This process was previously handled manually which requires a lot of human intervention. This is impractical if the schema is large as they contain large number of attributes to be matched. Hence, researchers have proposed to develop a model to solve the problems of schema matching without manual efforts and which is generic for all the
domains. But, this mission cannot be claimed to be completely successful as it was just implemented semi-automatically rather than fully-automatic model. Although we have several approaches which are claimed as generic, none of which performs the best under all circumstances. So, continuous efforts are in progress for developing a fully-automatic and a generic model. Thus, as part of our project, we have implemented a holistic approach which implements both the element-level techniques and instance-level techniques in order to find a match between the data across Gulf of Mexico.

Consider an example of bookstores Barnes and Noble, E-bay with the following schemas.

\( \text{Barnes\_Noble(Title, Author, Keywords, Publisher, Price, Format, Subject)} \)

\( \text{E\_bay(Book\_Name, Pub\_Name, Cost, Dom }\ove{\text{Fmt\_Type, Writer, keywd,)}} \)

Both the above mentioned schemas describe the database design of two famous bookstores and their schemas are not similar even though they are meant for the same business because they are liable to their own company standards. This would create confusion for an end-user who would like to purchase a book by exploring the various online bookstores. These are the type of scenarios where schema matching applications would solve these problems by matching the schemas against each other. These applications can match all the schemas and provide a match so that the end-user can make his decision. The output that is generated by implementing schema matching concepts is shown in Figure 2.1. From the figure, you can observe that each of the attributes of one schema matching with the other, it is not necessary that this happens with all the schemas.
From the above figure, you can observe that the two schemas are logically matched against each other by utilizing the matching techniques. The various schema matching techniques will be explained in the next section.

### 2.1 Approaches of Schema Matching

In this section, we will clearly explain about the various approaches of schema matching and their further classifications. To match the objects of the schemas, either instance-level or schema level match needs to be performed by using an algorithm based on the problem definition [Do 2005]. There are several methods to solve a matching problem, i.e. either by using an individual matcher which mainly computes a mapping for match criteria or we can even use a combination of these individual matchers together which is commonly termed as hybrid matchers. The diagram depicting the classification of the schema matching approaches are shown in Figure 2.2
Figure 2.2: Classification of Schema Matching Approaches [Rahm 2002]

Nowadays, due to the rapid development of many matching algorithms, many models are even trying to reuse the output that is generated from other matching algorithms so as to reduce complexity and time overhead. Hence, there is no particular approach to solve a problem rather than deciding based on the problem and the information that is available for us from the problem. All these approaches hold good for almost all the schemas (SQL, XML, OWL, DDL) because these methods mainly rely on the information that is present in these schemas rather than their representation. The overall descriptions of each of the matchers are as follows:

2.1.1 Schema-level Matchers

_These matchers consider schema information but not instance information._ The information that is available from the schema is name, description, data type, relationships, constraints and schema structure. We can divide schema-level matchers into two types based on their granularity i.e. element-level and structure level, which contribute to methods like constraint–based approaches and linguistic approaches.
2.1.1.1 Element-level matching

This method will perform a one-to-one, one-to-many and many-to-one match among all the attributes in the schema to obtain the best match among the schemas. It’s more of an atomic level where the attributes or column names in XML or relational schema will be matched. It can also be extended to coarser higher level elements like relational tables, cardinalities etc [Philip 2001]. This matching algorithm can be implemented similarly to relational join processing.

2.1.1.1.1 Linguistic based approaches

A linguistic approach uses the names and their descriptions to find a match correspondence between the other schemas. The two basic approaches in element-level matcher are:

(a) Name Matching:

This method will find a similarity between the attribute names of the schemas by many ways like name similarity, prefix, postfix, synonyms, hyponyms etc.

(b) Description Matching:

Many schemas are provided with brief description mainly referred as a comment in natural language to best describe the schema names. We can obtain the keywords from these comments and find a match among the elements based on the description provided.

2.1.1.1.2 Constraint based approach

Schemas usually contain constraints to define data types, values, indexes, cardinalities, relationship types etc. We can thus perform a match by utilizing these constraints from the two schemas. Some of the structural relationships can also be
claimed as constraints like foreign-key relationships, inter-schema correspondence etc. All these constraints can be matched based on the structural level matchers.

2.1.1.2 Structure level matching

This method will match the combination of elements in a structure. In this type of match, there could be some ideal cases where the entire structure of one schema will match with the entire structure of the other schemas and there could also be some matches where only some components of the schema do match with the other schema. Some of the common structures should be placed in the library that could be accessed and reused by the matching algorithms for some complex cases rather than doing it from scratch. Global cardinality n:m can be obtained using structural matching. Many schema structures are hierarchical, thus top-down approach is considered to be less expensive rather than bottom-up approach and hence this approach will be implemented [Do 2005].

2.1.2 Instance-level Matchers

Instance based matchers will be matched on the instance data rather than the schematic information. This match is considerably of great value when it is combined with schema level-matchers [Philip 2001]. The main problem that arises in these types of matchers is the quality of the data; the data should be cleansed before it is matched with other instances. Almost all the approaches described for schema-level matcher holds good for instance-level matchers as well, indeed there are additional approaches especially meant for these methods. They are described as follows:
2.1.2.1 Linguistic Characterization

This can be used for text attributes based on information retrieval techniques. We can extract the keywords and calculate their frequencies and identify the best match among them. Consider an example with attributes First_Last_Name, Full_Name, EmpName; we can come to a conclusion that First_Last_Name can be matched to Full_Name.

2.1.2.2 Constraint based characterization

This method is considered for structured data like numerical and strings. We can assign these columns with some ranges and averages for calculating the patterns. The columns which follow this rules are zipcode (which is mostly restricted to 5 digits), telephone numbers (with the format XXX-XXX-XXXX) and SSN (have a basic format XXX-XX-XXXX) etc. Most of these elements follow this rule in order to find a match.

2.1.3 Hybrid Matchers

This matcher will combine several matching approaches in a fixed way based on certain heuristics. It provides better performance as compared to the matching algorithms which usually work with a single approach. Since many of the intermediate passes are bypassed and can either be ignored if they are found to be less relevant, this approach yields better performance when compared to using a single matcher. Many of the approaches like CUPID, Similarity Flooding and SEMINT are currently using this hybrid matching approach.

The main drawback with this approach is it cannot be extended or improved so as to incorporate new match criteria, new information sources etc. This problem is addressed and it can be resolved while using a composite matcher.
2.1.4 Composite Matcher

This method combines the results from the independent matchers. They can use a particular matching algorithm based on the application domain that is available in the repository. We have several matchers available to match in a particular scenario; hence the selection and their strategy can be defaulted to perform this automation directly. Due to its flexibility and extensibility, we can generate and develop a generic match method for specific application domains and schemas. COMA and COMA++ are the only approaches that are currently utilizing this matcher [Philip 2001].

2.2 Types of Schemas for Matching

These are three basic types of schemas that need to be addressed for the problem of matching. A pictorial representation of an example of purchase order is well explained in these three schemas in the Figure 2.3.

![Figure 2.3: Sample schemas for purchase order [Do 2005.]]
A brief description of each of them is as follows:

2.2.1 Structural Query Language (SQL)

*SQL is defined as a schema in relational databases in order to query and manipulate data stored in those schemas.* This schema contains a set of tables like *PO* and *ShipTo* and it is comprised of several attributes, and some attributes might also have correspondence with a column in another table to maintain referential constraints [Do 2005]. Some values are also stored in these schemas for querying which are considered as instances. Schema matching tools are used to co-relate these SQL schemas using their methodology. Almost all the methods initially consider SQL schemas as their base research point for schema matching and extend their research to XML and OWL as well.

2.2.2 Ontologies

*This is a study of a group of objects that exist in a particular domain, their relationship with other object, etc* [Wikipedia 2010]. Basically, ontology has its roots from the branch of philosophy called metaphysics which deals with the reality of the nature. The main components included are instance, attributes, events, relations etc. Schema matching is a necessary step to identify a correspondence between two Ontologies.

2.2.3 XSD (XML Schema Definition)

They are used to illustrate the structure of the XML documents which contains data that is accessible over the web. XSDs mainly consist of elements (P0), attributes (ShipTo, BillTo, etc), and types (string, decimal etc). They even have a complex nested sub-element structure or they are even used to specify the data types of the attributes [Do
2005]. As observed in the Figure 2.3, we can find the values only for elements, the attributes and types are used to describe nature of the attribute rather than a value.

We have schema matching methods like COMA++ which has mainly focused in identifying a match between the XML documents [Reynold 2009]. As part of their research they even proposed a novel model called “block tree” in order to identify a match between the XML documents.

2.3 Project Data

We have obtained the data from few of the systems of Gulf of Mexico Coastal studies for this project. This domain has been considered so that it could be useful for the Corpus Christi officials to identify the hurricane threat before hand by matching the water-level information from various systems and take necessary action. We have primarily focused on the data from the major observing stations along the coastal region.

2.3.1 TCOON (Texas Coastal Ocean Observation Network)

It’s a collaboration of Conrad Blucher Institute of Surveying and Science at Texas A&M University-Corpus Christi in 1989 which installed several observing stations along the coastal region to measure the water level. In 1991 it was named as TCOON with an official webpage (TCOON, http://lighthouse.tamucc.edu/TCOON/HomePage) [TCOON 10]. TCOON is managed by several higher officials in Texas. The water-level measures and other parameters are available via cellular telephone, packet radio, and Geostationary Operational Environmental Satellite (GOES). These measures are captured for every 6 minutes. A view of the observing stations located at the coastal region is attached in Figure 2.4.
These systems are primarily focused to:

- Provide water-level and meteorological measures for every certain amount of time.
- Prepare the affected communities for evacuation during hurricane or tropical storms.

Other than the basic purpose, it serves its best for providing information regarding oil-spill, oceanographic and environmental research, coastal engineering studies etc. They not only capture water-level information but also provide wind speed, wind direction, air temperature etc.

Since the measures are captured at every 6 minutes, the data is humongous. Hence, several projects are developed to maintain the data like Nueces Bay Salinity Project, Conrad Blucher Institute for Surveying and Science.
2.3.1 TABS (Texas Automated Buoy System)

During 1994, the General Land Office (GLO) of Texas has instructed the Geochemical Environmental Research Group (GERG) at Texas A&M to develop a system that provides real-time observations of surface contents and water temperatures at the Texas coastal regions. The official webpage for TABS is http://tabs-os.gerg.tamu.edu/tglo/ [TABS 2010]. The buoys are the measuring stations in TABS. This system has the high capability of identifying the path of the spilled oil in Texas. An image of the buoy which captures both the surface currents and the water levels are showed in the figure below.

![Diagram of TABS buoy models](image)

Figure 2.5 Schematic diagrams of the TABS buoy models

These buoys are installed in several regions across the coastal areas in order to predict the measures and also identify the path of the oil-spill. This is the most effective public resource for coastal studies for the following reasons.
• All the TABS buoy data is available for access to everyone by a user-friendly webpage.

• It supports research to improve the reliability, operational range of the research.

• It also encourages other projects to consider the data from the TABS sources.

These buoys capture data for every 30 minutes. Tabs buoys are located across Texas like Galveston, Arkansas, Brownsville etc. These observations are being viewed everyday by boaters, companies, fishing vessel, companies etc. TABS was proved successful in real spills and real drills like Buffalo Marine Barge oil spill to identify the water currents and find the path of the oil spill [Kelly 1998].

Apart from these two mentioned above, we have used data from other coastal region observing stations in random.

2.4 Tools developed as part of the project

Holistic Schema Matcher is a tool that has been developed as part of the research. In this application, we are mainly focusing on the holistic schema matching which combines the matching of both instance-level matching and element-level matching. We have used JAVA for our basic implementation. This application has the capability to match three schemas at a time and display the matches among them. A green line will be displayed for element-level matching and a pink-line will be displayed for instance-level matching.

2.5 Scope of the Project

This project mainly uses the data from the two measuring stations like TCOON and TABS and also considers data from other observing stations on random. In this project, we have made our efforts in identifying a match between these measuring
stations by closely looking at them and implementing the schema matching concepts. The TCOON schema is a large one when compared to TABS data. This data is primarily considered for matching since it could be useful for other existing projects as well.

**2.5 Java Swing Jargon**

Swing is a widget toolkit, in JAVA, for generating Graphical User Interfaces (GUIs). In this project we have used NetBeans IDE which is a free and open source environment which supports the execution of Java programs. Other alternatives are JCreator, Eclipse etc. The GUI can be built in an easier method in NetBeans by using drag and drop options. This will generate the code automatically which allows us to focus more on the application rather than the look of the application. The main characteristics of Swing are look and feel, accessibility, drag and drop etc. It is known as a library extension of AWT which is much better when compared to AWT and it can also be used to build standalone desktop applications in a much simpler way [Swings 2010]. The main components of Swing that are used in the application are as follows:

1. **JFrame:**

   It is a window which contains all user-defined components like buttons, toggles, title, border etc. Any component that has to be included in the window should be included in the content pane.

2. **JLabel:**

   It is a subclass of JComponent. It is mainly used to display the text instructions on the GUI. In our desktop application, we have used JLabel to provide the status of the matching.
3. **JButton:**

These are simple buttons. These can be used to allow user to trigger different events. In our application, by clicking on different buttons, the user can select files and get the matching results displayed.

4. **JDialog:**

It is basically used to display a pop-message which indicates a temporary notice [Schildt 05]. In our case, it is used to display the pop-message “Only .xls and .txt formats are allowed”.

5. **JXL:**

Java Excel API is a recent functionality which is used to read the data from Excel spreadsheets. This is the most important object used in our application since we have considered excel as one of our input formats. This component allowed us to read the data from excel, pass them as inputs to few of the matching algorithms and then display a result if a match is found.
3. SYSTEM DESIGN

Holistic Schema Matcher is a JAVA application which can match two or three schemas at a time and generate the output based on element-level matching techniques, instance-level matching techniques, and a combination of both. This is a desktop based application which is primarily developed for the datasets that are obtained in the Coastal studies like TCOON, TABS etc. These schemas can be matched against each other so that this information would be useful for the city officials in order to take rescue actions during hurricanes or tropical storms. The schema names are generic because each of these schemas is defined by a database administrator whose schema names vary from the schema names of the other dataset. Hence, this would create havoc in terms of matching. In such cases, this tool would play a better role in matching those schema names and also match those schemas based on the instances and provide good input for the officials regarding the meteorological data.

The inbuilt concept of match is to consider two or more schemas as input and produce a semantic matching between the elements-level and/or instance-level of these schemas with each other. Although we have several matching algorithms like CUPID, COMA, ARTEMIS etc which are currently in existence, we have made our efforts in developing a holistic matching application which could take into consideration both of the key matching techniques (element-level and instance-level). The application is developed in such a way that it can accept inputs in formats like excel (.xls) and text (.txt). The outputs for each of these schemas are displayed with the help of the green line for element-level matching and magenta colored line for instance level matching. It
is set up in such a way that, there is a possibility to view the matching’s individually as well as together which is known as holistic matching approach.

3.1 System Requirements

Running this application requires the following software and tools:

JAVA JDK 1.6 and Netbeans 6.9.1

3.2 System Architecture

This is a hybrid matcher application which combines both element-level matching technique and instance-level matching technique. This application has to be executed using NETBEANS 6.8 version or higher. The Graphical User Interface (GUI) has the option of browsing for three files with excel (.xls) and text (.txt) formats. It has the capability of matching three schemas at the same time. The architecture follows a holistic approach of schema matching. The architecture of the system is shown in the Figure 3.1.
The architecture mainly consists of five components. Each of them is explained below briefly.

1. **Inputs:**

   This component will deal with the inputs provided by the end-user to this application. The inputs should be either in excel (.xls) or text (.txt) formats. It can accept and perform the matching operation for three schemas at a time. We have three fields in the GUI named Schema 1, Schema 2, and Schema 3 where you can browse for a particular file that needs to be matched and click the perform matching button in order to execute the process. Few of the precautions that need to be taken regarding input files are:

   - File formats should have an extension .xls (MS Office 2003) instead of .xlsx (MS Office 2007).
   - In case of text files, the data should be in comma separated file.

2. **Element-level techniques:**

   This is one of the most important functionality of the system. It mainly focuses on the schematic information. The schematic information constitutes the element names, data types, descriptions, formats, relationships etc. This technique contains all the logic for performing element-level matching. The key techniques that are used for implementing element-level matching are string-level techniques, constraint-based techniques, language based techniques and linguistic resources. A brief description of each of them is explained.
a. **String-based techniques:**

As the name suggests, it is mainly based on matching the element names and description of the schemas. Here, we try to find a match for a sequence of the set of words. The underlying concept is that if we can observe more similarity in the strings then it is more likely to be matched. The basic tests that are performed in these techniques are as follows:

*a. Prefix:*

This test takes two strings as input. If second string starts with the first string then first string is said to be the prefix of second one. It seems to be successful at times when the first string is an abbreviation of the other. This function returns a binary value (True or False). If the output is true then we match else we ignore it.

Ex: Consider auth and author where both of them are referring the author for a book.

*b. Suffix:*

This test is similar to prefix test function where it searches for a second string which ends with the first string. It returns a binary value where True is considered for matching. It is almost similar to the prefix function with a slight change.

Ex: Consider number and enumber where both of them are referring to the employee number.
c. **Edit distance:**

This algorithm computes the number of operations (adding/deleting/replacing a char) required to transform first string into second string. The output of this function is used in a formula to find the similarity of the strings. That formula will produce an output in the range of 0 to 1. The formula used to calculate the similarity is:

\[
\text{similarity of two strings} = 1 - \frac{\text{editDistance}}{\text{avgStringLength}}
\]

The edit distance of tkt and ticket is 0.5. This function returns a value, and we decide the match based on that value.

b. **Language based techniques:**

This also deals with the names and textual descriptions of the schemas. It is mainly implemented using Lemmatization and Tokenization [Lemmatizer 2010].

a. **Tokenization:**

This is the process of breaking up a text into valid words, synonyms or phrases and matching these against the elements in the other schemas. Hence, we can come to an understanding that a text is composed of meaningful words which could be the element names of the other schema.

Consider an example of “date_time” element, this element can be broken down into date and time and these values can be matched against the other schema which might contain date and time.
b. **Lemmatization:**

This process gathers all the tokens obtained from tokenization and manipulate them to find all the available basic forms to match the other schema [Attivio 2010].

For example, ‘Dates’ of one schema can be matched with ‘Date’ of another schema using this method.

c. **Linguistic Resources**

This technique will build a knowledge base of the common words, synonyms of some frequent terms used in coastal studies systems to match the elements across other schemas. A domain knowledge thesaurus is created which is explained below.

a. **Domain knowledge thesauri:**

It contains some terminology that is often used in these measuring systems. We have created an extensive knowledge base which clearly explains the meaning of a specific terminology based on which we can decide whether one column matches the other or not.

For example, the elements for the TCOON stations are like wsd, wdr etc.

So, we have created a knowledge base describing that wsd= wind speed, wdr=wind direction and so on.

These are the various methods that we have implemented for identifying a match based on element-level in this system [Do 2005]. Whenever a match is found among the element names of the schemas using any of the above techniques, a
green color line is displayed from one element of schema 1 to another matched element of schema 2.

2. **Instance level matching techniques:**

Instance data will provide more information from the contents of the schema elements. Almost all the methods that have been used for element-level matching will hold good for instance level matching as well. Most of the instance-level matchers reuse the results that are obtained from the previous algorithms.

In this project, we have used query occurrence matrix to calculate the match among the instance data. Query occurrence matrix (QOM) together with Estimated Mutual Information matrix (EMI) will represent the relationship between the attributes in the schemas [Jiying 2004]. Suppose the number of attributes in the schema 1 is considered as N, the number of elements in schema 2 is considered to be M, and then the matrix will be represented in the form of $QOM_{N \times M}$. This formula can be used for a pair of schemas at a time. The first step is to calculate the QOM and then based on the values from the QOM, we will calculate the EMI to match the attributes among each other.

- Calculate the QOM: This is calculated by using this formula, which is an intermediate step in identifying the match between the attributes.

$$Occurrence\ Matrix \ (A_i, B_j) = \left( \frac{m_{i+} \cdot m_{+j}}{M} \right)^2$$

----------Equation (1)

where $M=\sum_{i,j} m_{ij}$

$$m_{i+}=\sum_{j} m_{ij}$$
and $m_{ij} = \sum_i m_{ij}$

- Now, we calculate the Estimated Mutual Information from the QOM and build the EMI matrix. In EMI matrix if a cell contains largest value row wise as well as column wise, a match is considered to be found among the corresponding attributes.

$$EMI(A_i, B_j) = \frac{m_{ij}}{M} \log \frac{m_{ij}}{m_{i+} \cdot m_{+j}}$$

-----Equation (2)

The EMI matrix of a particular case is shown below. The Figure 3.2 also explains how an attribute matches with the other.

Figure 3.2: Results generated based on the formulas

As represented in the figure and based on our implementation strategy, the value which is highest row-wise as well as column-wise is considered to be a match. So, we can observe that attrib1 matches with the attrib8, attrib2 matches with attrib9 and so on. But, if we can observe more clearly, the attrib3 doesn’t have any match with any attributes, since it doesn’t have a value which is largest row-wise as well as column-wise. Hence, whenever a match is found in the occurrence matrix,
we display the match by generating a magenta color line from one attribute of one schema to another attribute of the second schema.

Based on the results, we are able to find a maximum number of instance-level matching in the schema since, it is not claimed that an attribute would have a highest value with at least one attribute of other schema. A match among the schema attributes is possible only when that value is largest row-wise as well as column-wise. If the value is largest row-wise but is not largest column-wise even then the attributes won't be considered to be matching each other. This is the strategy that we have implemented in order to solve the instance-level matching problem.

3. **Holistic Schema Matching**

Holistic Schema matching is defined as the combination of element-level matching technique and instance-level matching techniques [Priyanka 2009]. The method described in the paper [Priyanka 2009] mainly uses Kang and Naughtons method for instance level approach and Li and Yangs method for element level approach. In the instance level approach, a dependency graphs are constructed for both the source schema and target schema. A match is considered with an optimal score obtained from the Euclidean distance of the vertices. In element level approach, the match process calculates linguistic similarity and structural similarity in star schema. The first step is to find the fact candidate by co-relating the other candidates in the source schema, the second process is to match each of the attributes of the target schema to this fact candidate and calculate the score and binary schema tree is constructed. The match is decided based on the similarity
metrics of each of the attributes of the schemas. The Holistic approach in the paper [Priyanka 2009] is implemented by improvising the dependency graph metrics and extending the process to relational schemas as well.

The method implemented in this project gives us more flexibility in combining the results from the other methods. This project will append the results that are obtained from element-level as well as instance-level by using a linked list function.

4. GUI

This is the user interface and the output screen which displays the matching results after the user provides inputs. In this tool, we have provided an option to observe element-level matching outputs, instance-level matching outputs and holistic matching outputs. We have provided radio buttons so that the end-user can select the output that is needed.

- If the element-level option is chosen, then you can view the results that are obtained by using the element-level matching techniques by displaying the match among the elements with a green line.
- If the instance-level option is selected, the matching results are obtained by using the occurrence matrix which is represented by magenta line.
- If the holistic level option is selected, a matching with a combination of both the element-level and instance-level are formed and displayed on the GUI. This is implemented by using a linked list.

Apart from the above important blocks of the system, we have some more options like:
• **Perform Matching:**

  This push button will execute the code and generate the results for the given input schemas.

• **Next:**

  It will show the results for the next set of inputs. For example, if we have taken three input schemas into consideration and the output of first two schemas is displayed on the GUI, we click this button in order to display the results of the next set of schemas.

• **Prev:**

  This is similar to the Next button, except it will show the results of the previous sets of schemas.

• **Label:**

  It will display the status of the current files that are being shown in the display. For example, it will show that “Comparing file1 with file2”. This is the message that will be displayed while displaying the results for file1 and file2.

### 3.3 Flowchart of Holistic Schema Matcher:

This flowchart will give a brief idea of the execution of holistic schema matcher application. Consider three schemas S1, S2 and S3 as inputs to the application. Schemas in the form of .xls or .txt are taken as inputs and are executed by clicking the *Perform Matching* pushbutton which performs each matching algorithms simultaneously and the match is displayed based on their matching criteria. This button will perform all the matching techniques at the backend and stores the results in a pair of parallel linked list. The output will be displayed according to one of the
selected options among the three matching techniques. We can also backtrack the results of each of these pair of schemas by using the Next and Prev button available.

You can also view the current status of the matching in Label button. This button will display a status message like “Comparing schema 1 and schema 2”. The Figure 3.3 will clearly explain the sequence of the flow of the application.

![Flowchart](image_url)

Figure 3.3: Flowchart depicting the execution of the application
3.5 Demo of the Tool:

After setting up the JAVA environment and also installing NetBeans 6.9.1 (you can also use NetBeans 6.8.1). The *schemamatcher* JAVA code could be opened in NetBeans from ‘open project’ option in file menu. You should make sure to include jxl package into your JAVA library otherwise your code wouldn’t be compiled successfully. You can execute your project by either clicking the run button or clicking F6 in NetBeans, and the GUI will be displayed as follows:

![Figure 3.4 The GUI for Holistic Schema Matcher application](image)

It can be observed from the figure 3.4 that we can browse for an input file at any location and their locations are currently set to default locations. Once all the schemas are selected that need to be matched, we can click the Perform Matching button, and then
select the matching techniques that need to be implemented from the three options to display the output.

Figure 3.5 represents the inputs selected from TCOON, TABS, Meteorological Survey data and then executed the application by clicking Perform Matching button having Element Level Matching techniques selected from the three options available. The green lines are drawn from one element of the schema to another element of another schema.

![Figure 3.5: The output generated from Element level matching technique](image)

In this diagram, it can be observed that each element is matched to their corresponding matching element in the second schema. This result is generated by using string-matching techniques, language-based techniques, constraint-based techniques, and linguistic resources. A knowledge base has been created for the columns of the TCOON
data since they are not very clear. This matching is performed on MS.xls schema and Test_data_TCOON.xls schema. You can also view the matching of the other schemas by clicking the Next and the Prev button.

Figure 3.6 represents the output that is generated from instance level matching techniques. This output will show magenta color lines which will connect each attribute of one schema to the attribute of the other schema. The figure displays only few matches of the instances among the schemas because in occurrence matrix, according to the literature used as help material, a match is found only when a largest value is claimed from row-wise and column-wise. That is the reason we have limited number of matches in instance-level matches.

Figure 3.6: The output generated from instance-level matching techniques
Now, we will show the holistic level matching results by selecting the last option available. This will generate the outputs from both the methods using a linkedlist. The output is shown in the Figure 3.7 as follows.

Figure 3.7: The output generated from Holistic schema matching

In the above figure, you can notice both the green lines as well as magenta lines being displayed as output for holistic schema matching. Figure 3.7 represents the output generated by schema 1 and schema 2.
This Figure 3.8 will depict the matching between schema 2 and schema 3. It depicts the holistic schema matching between them.

Figure 3.8: The output generated by schema 2 and schema 3
The Figure 3.9 will show the matching between schema 3 and schema 1.

![Image of schema matching application](image-url)

**Figure 3.9: The output generated by schema 3 and schema 1**

Hence, this clearly explains that this application can match three schemas using element-level techniques and instance level techniques as well.
4. TESTING AND EVALUATION

4.1 Testing

Testing is defined as the process of verifying and validating that the software product has met all its functional requirements. Testing can be performed at any phase of software development life cycle model, and it is carried out only after the requirements have been completely agreed upon. This application has been tested at various levels and they are as follows:

- **Unit Testing:**
  
  We have tested the individual blocks of code for each of the matching techniques and made sure that they are meeting the requirements.

- **Integration Testing:**
  
  We have integrated all the components of the application and then tested as a whole and verified that the results generated are same as the one when tested individually.
The main test cases that have been tested are as follows:

**Test Case 1:** To validate that the three matching techniques are performing as expected using .xls formats as inputs.

**Observed Output:**

![Figure 4.1: Results obtained from Test Case 1](image)

We have matched these set of schemas manually and they seem to appear the same as the one shown in the above figure 4.1. It is clearly observed that three schemas with the same input format is being matched without any problems.
Test Case 2: To validate that schema matching is also performed between two different types of formats.

Observed Output:

Figure 4.2: Results obtained from Test Case 2

The matching is working successfully even for different sets of inputs. The one which is highlighted is a text file and is shown in Figure 4.2. It can be induced that the application is matching for a different set of inputs.
**Test Case 3:** To validate that the application is working fine for boundary value analysis. In this case, we have limited our input formats to .xls and .txt. So if we happen to provide other sorts of input, then we should notice a pop-window.

**Observed Output:**

![Application Interface](image)

Figure 4.3: Results obtained from Test Case 3

Hence, this shows that the application is tested for almost all the test cases as we have tested with all the in-bound values and out-of-bound values in Figure 4.3. This gives us the verification that it is working fine for out-of-bound values too.
4.2 Results Analysis:

4.2.1 GCOOS Domain

The results obtained by considering the GCOOS domain are tabulated, three schemas are taken as inputs and match is performed by using each of the matching techniques. The schemas and the attribute count is as follows.

Meterological Survey data: 7

TCOON (Texas Coastal Ocean Observing Network ) = 26

TABS ( Texas Automated Buoy System) = 14

- The element level matching techniques are performed on these set of schemas and the results are tabulated in Table 4.1

<table>
<thead>
<tr>
<th>Number Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(7), S2(14)</td>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>S2(14), S3(26)</td>
<td>8</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>S3(26), S1(7)</td>
<td>6</td>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.1: Results obtained from element level matching techniques

- The results obtained from the instance level matching techniques are tabulated in Table 4.2

<table>
<thead>
<tr>
<th>Number Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(7), S2(14)</td>
<td>3</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>S2(14), S3(26)</td>
<td>4</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>S3(26), S1(7)</td>
<td>4</td>
<td>3</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4.2: Results from instance level matching techniques
• The statistics obtained from the output of the Holistic Schema Matching techniques is shown in Table 4.3

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(7), S2(14)</td>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>S2(14), S3(26)</td>
<td>9</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>S3(26), S1(7)</td>
<td>7</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 4.3: Results from Holistic Schema Matching Techniques

These are the results obtained the data obtained from GCOOS domain.

4.2.2 Books Domain

The statistics of the results obtained by using the book schema information is described in the table as follows. Three schemas are considered from Books domain and the tests are performed. The number of attributes of these schemas are as follows.

Schema 1: 4
Schema 2: 5
Schema 3: 4

• The results obtained from element level matching techniques are shown in Table 4.4

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(4), S2(5)</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S2(5), S3(4)</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S3(4), S1(4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.4: Results obtained from element level matching technique
• Statistics from the results of the instance level matching techniques are represented in Table 4.5

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(4), S2(5)</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S2(5), S3(4)</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S3(4), S1(4)</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.5: Statistics from the output of Instance level matching

• Results obtained from holistic matching techniques are shown in Table 4.6

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed Schema S1</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(4), S2(5)</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S2(5), S3(4)</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S3(4), S1(4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.6: Results obtained from Holistic matching techniques

These are the results obtained by considering three schemas from the Books domain

4.2.3 Movies Domain

We have considered schemas from movies database to analysis the results obtained from the tool. The number of attributes of the schemas is as follows:

Schema 1: 3
Schema 2: 8
Schema 3: 4

• The statistics obtained by implementing element level matching techniques is shown in Table 4.7
<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed S1</th>
<th>Missed Schema</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(3), S2(8)</td>
<td>3</td>
<td>0</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>S2(8), S3(4)</td>
<td>4</td>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>S3(4), S1(3)</td>
<td>3</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.7: Results obtained from element level matching for Movies domain

- The results obtained from instance level matching techniques are tabulated in Table 4.8

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed S1</th>
<th>Missed Schema</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(3), S2(8)</td>
<td>2</td>
<td>1</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>S2(8), S3(4)</td>
<td>3</td>
<td>5</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>S3(4), S1(3)</td>
<td>2</td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.8: Results obtained from instance matching techniques for Movies domain

- Results obtained from the Movies schema by performing holistic matching is shown in Table 4.9

<table>
<thead>
<tr>
<th>Number of Attributes</th>
<th>Matched</th>
<th>Missed S1</th>
<th>Missed Schema</th>
<th>Missed Schema S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1(3), S2(8)</td>
<td>3</td>
<td>0</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>S2(8), S3(4)</td>
<td>4</td>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>S3(4), S1(3)</td>
<td>3</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.9: Results obtained from holistic matching for Movies domain

These are the results obtained from the schemas taken from Movies domain.
5. FUTURE ENHANCEMENTS

A holistic schema matcher application served its purpose of identifying a match between the schemas of the various measuring stations installed on the coastal studies of the Gulf of Mexico. We would like to enhance this project to accept input formats other than excel and text (mainly comma separated file). Basically, we would also like to tune the instance-level matching algorithm to match more attributes than the existing one. The most important enhancements are as follows:

- Also increase the input schemas to more than three and improve the look of the GUI with the latest components of the time.
- Can access any sort of delimited file like pipe-delimited, colon separated, space delimited file etc.
- Extend the functionality to be accessed to all the domains.
- Currently, we are performing matching only for SQL; hence we would like to enhance it to XML and OWL matching as well.
- Utilize the other matching techniques like structured level and information retrieval techniques etc to perform schema matching.
- We would also like to develop an application which doesn’t need any human intervention, since this process needs at least a minimum percentage of human intervention.
- Increase the functionality of the application by improving the concepts that are currently used in it.
- Generate a statistics to analyze the results that are obtained from each of matching techniques.
Apart from the above, we would also like to install these applications in almost all the most necessary places like e-commerce, semantic web query processing, data integration projects, etc.
6. CONCLUSION

This project has been developed to match the measuring stations (like TCOON, TABS etc.) of the coastal studies located across the Gulf of Mexico. Hence, we have focused on this domain in order to evaluate our approach. We have implemented the element-level and instance level techniques in order to achieve a match. Holistic Schema Matcher Application is a user-friendly JAVA based application designed to match the schemas through different techniques. Currently, schema matching is rigorously needed in almost all the applications and many researches are in place. Various Element level matching techniques like string based, constraint based, linguistic resources, language based and Occurrence matrix is used for performing a match for instance data.

Holistic Schema Matcher Application can match three schemas at a time and a match is obtained using the instance level and element level matching techniques. Several options like element level matching results, instance level matching results and holistic level matching results are present to display each of the matching results simultaneously as well as individually.
BIBLIOGRAPHY AND REFERENCES


APPENDIX –A

• Schema
  
  Schema is the database structure that is supported by database management systems. It represents the tables that are presented in the database in the form Tablename (attrib1, attrib2,…)

• Schema Matching
  
  This is the process of finding the relevant matches of the attribute names and instances of the schema using element-level techniques and instance-level techniques.

• Ontology
  
  It is the logical representation of a particular object, its existence and characteristics. It mainly comes under the field of psychology.

• Lemmatization
  
  This is one of the techniques used for element-level matching where one form of word is transformed into several forms by inserting, deleting, updating the words and finding a match between the other attributes.

• Tokenization
  
  It is the process of breaking or splitting a phrase or a large word with small meaningful words and then finding a match with them.

• Element-level
  
  This is one of the techniques that we have used in our project, this technique will perform a match based on the element names that is available from the schematic information.
• Instance-level

This is one of the second techniques that used to generate the results for the schema matching. It mainly deals with the instance data rather than the column names.

• Holistic Schema matching

It is a process which combines the results that are obtained from both the element-level techniques as well as instance-level techniques.

• TCOON

Texas Coastal Ocean Observation Network

• TABS

Texas Automated Buoy Systems
APPENDIX – B

We have considered data from TCOON, TABS. The schemas for each of these are described below.

TCOON:

This schema contains 26 attributes and 100 records of instance to perform the testing.

The Schema for TCOON is as follows:

$$TCOON(\text{Date, pwl, bw, harmwl, atp, wtp, wsd, wgt, wdr, bpr, wsd2, wgt2, wdr2, swh, pwp, vlx, vly, vlz, vln, vlu, vles, ssz, hhw, lh, s})$$

Where pwl= primary water level

Harmwl = harmonic water level

Atp = air temperature

wtp = water temperature

wsd = wind speed

wdr = wind direction

wgt = wind gusts

bpr = barometric pressure

vlx = velocity x

sal = salinity.

ssz = signal strength z

These are the basic abbreviations of these attributes.

TABS:

This schema contains 13 attributes and the schema is shown below.
TABS(DATE, TIME, VEAST, NORTH, AIRT, ATMPR, GUST, COMP, TX, TY, PAR, RELH, Wind Sp, Wind Dir)

Where TX=Time X

    Comp=Compression

    RELH= Relative Height

and the attributes are almost similar to the TCOON attributes.

MS:

This contains the information of the meteorological survey and the schema is as follows:

MS(Date, Time, Wind Sp, Wind Dr, Wind Gt, Air Tmp, Baro Pr)

We have considered these three schemas and tested them. We can test them with any measuring stations of the coastal studies.
APPENDIX – C

Coding Details

The project contains about ten java files and total of 3000 lines of code. The remaining documentation will briefly explain how the code works and what are the data structures and algorithms used to get the desired results. There are five main files that contain most of the implementation. Those files are SchemaMatcherView.java, FunctionDefinitions.java, SnowballProgram.java, EnglishStemmer.java and SchemaMatcherAboutBox.java

a. SchemaMatcherView.java:

This file contains the code that runs behind Graphical User Interface (GUI). Any action on GUI like click on a menu, click on a button or writing text in a text field is handled in this file. In short this file tells the application that what to do when the “Perform Matching” button is pressed, what to do when a radio button is selected.

b. FunctionDefinitions.java:

This file contains the functions that implement the core algorithm of the application. The file even contains most of the implementation of Schema Matching as well. The function like readExcelFile(..) loads the excel file and recognizes the attribute names. This function also records the data-types and dataLengths of each attribute. Similarly a function named readTxtFile(..) does the same for text files. The function findMatched(..) contains the core flow of the match finder algorithm. This function calls many other helper functions in order to perform the matching. The function findInstanceLevelMatchings(..) finds the instance level matching. The function
calculateEditDistance(..) calculates the edit distance between two given strings that is used in Element-Level Schema Matching.

**c. SnowballProgrammer.Java and EnglishStemmer.java:**

These two files contain the core implementation of Lemmatization, and their functions are called from a function of the file FunctionDefinitions.java to find the basic forms of the strings (attribute name).

**d. SchemaMatcherAboutBox.java:**

This file contains the code for the about-box of the application.