A STATISTICAL DATA FUSION TECHNIQUE IN VIRTUAL DATA INTEGRATION ENVIRONMENT

Mohamed M. Hafez[1], Ali H. El-Bastawissy[1] and Osman M. Hegazy[1]

[1]Information Systems Dept., Faculty of Computers and Information, Cairo Univ., Egypt

ABSTRACT

Data fusion in the virtual data integration environment starts after detecting and clustering duplicated records from the different integrated data sources. It refers to the process of selecting or fusing attribute values from the clustered duplicates into a single record representing the real world object.

In this paper, a statistical technique for data fusion is introduced based on some probabilistic scores from both data sources and clustered duplicates.

KEYWORDS

Data integration, duplicates detectors, data fusion, conflict handling & resolution, probabilistic databases

1. INTRODUCTION

Recently, many applications require data to be integrated from different data sources in order to satisfy user queries. Therefore, it was the emergence of using virtual data integration. The user submits the queries to a Global Schema (GS) with data stored in local data sources as shown in Figure 1. Three techniques (GaV, LaV, and GLaV) are used to define the mapping between the GS and local schemas (LS) of the data sources[1], [2]. In our technique, we focus more on the GaV technique through defining views over LS. The defined mapping metadata determines the data sources contributing in the answer of the user query.

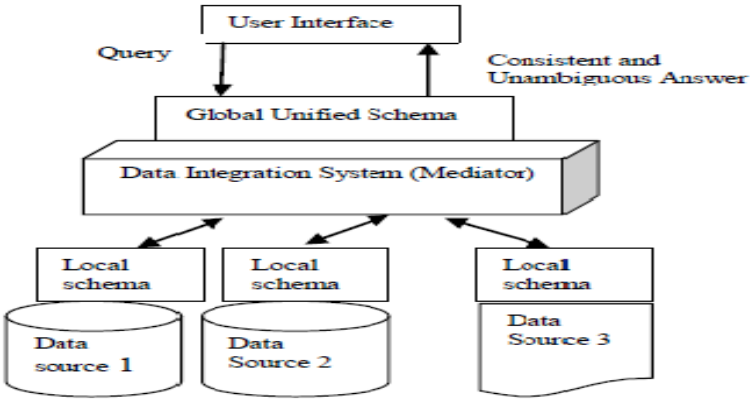


Figure 1: Data Integration Components[2].

DOI : 10.5121/ijdkp.2013.3503

In such an integration environment, the user submits the query and waits for the clean and consistent answers. To give the user such results, three steps should be processed in sequence: Schema Matching, Duplicate Detection and Data Fusion as shown in Figure 2.

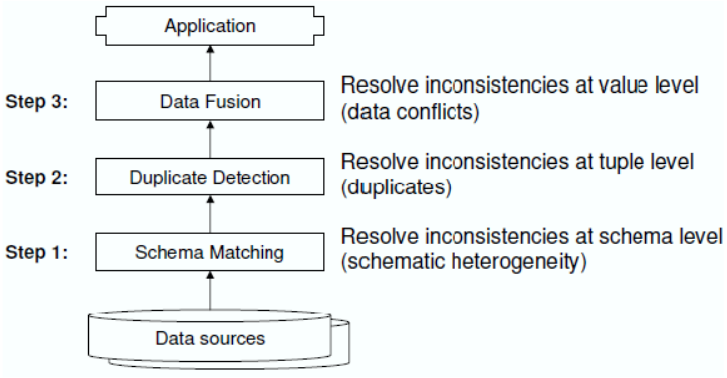


Figure 2: The Three Steps of Data Fusion[3].

Schema matching inconsistencies are handled in many research articles [4][5]. In step 2, Integration the answers from the contributing data sources, many duplicate detection techniques could be used to discover and group duplicates into clusters. Data conflicts arise after step 2, which should be handled before sending the final result to the user.

Many possible obstacles should be addressed in both step 2 and step 3. One problem that might arise in step 2 is if a set of records were considered duplicates incorrectly. Do we ignore these duplicates? Can we use some predefined metadata to help solving this problem? Thus, we have a challenge to find a smart way to improve the findings of such duplicates. Another important problem is that within each cluster of duplicates, some values might conflict between different records representing the same object. In order to solve such conflicts, should we ignore them and rely on the user to choose from them, or to avoid such conflicts from the beginning by putting some data sources preferences and choose based on them preventing any conflicts to occur? Or one last option is to resolve such conflicts in the run-time using some fusion techniques? Therefore, the challenge to be addressed here is to find the most appropriate way to overcome conflicts and complete the data fusion process.

This paper is organized as follows; in section 2 we mention the description and classification of the commonly used conflict handling. In section 3, the proposed technique for data fusion will be explained, which is the core of our work. Then, in section 4 our five steps data fusion framework will be presented. The conclusion and future work are presented in section 5.

2. RELATED WORK

Many strategies were developed to handle data conflicts, some of them were repeatedly mentioned in the literature. These conflict handling strategies, shown in Figure 3, can be classified into three main classes based on the way of handling conflicting data: ignorance, avoidance, and resolutions[6], [7],[8].

2.1. CONFLICT IGNORANCE. No decisions are made to deal with conflicts at all; they are left to the user to handle them. Two famous techniques to ignore such conflicts: **PASS IT ON** that

takes all conflicting values and passes them to the user to decide, and the other one is **CONSIDER ALL POSSIBILITIES** which generates all possible combinations of values, some of them don't have to be present in the data sources, and show them to the user to choose.

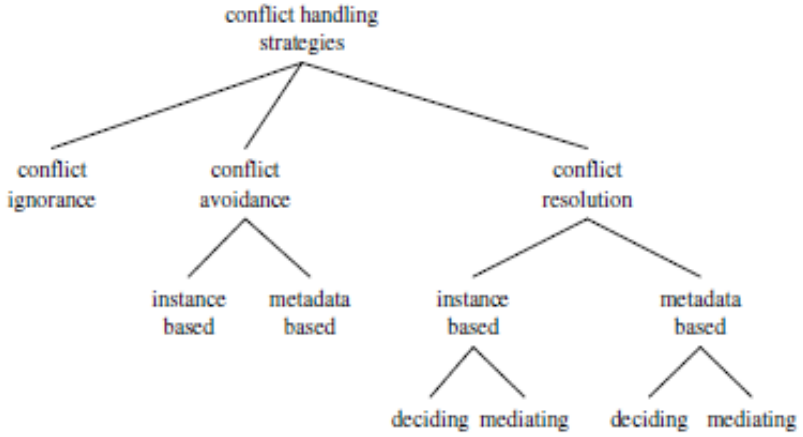


Figure 3: Classification of the Conflict Handling Strategies[6].

This strategy works in any integration environment and could be easily implemented. From the computational point of view, it might not be so effective to consider all combinations and give them to the user for choosing. From the automation and integration point of view, it is ineffective to involve the user in deciding on the good values because the user doesn't know about the structure of the integration system nor the organization and quality of data sources.

2.2. CONFLICT AVOIDANCE. Decisions are made before regarding the data values, which prevents any possibility of hesitation or making decisions about the values to be chosen in the data fusion step before showing the final result to the user. Conflict Avoidance can rely on either the instances/data sources or the metadata stored in the global schema to deal with such conflicts. Two famous techniques based on the instance are: **TAKE THE INFORMATION** in which only the non-NULL values are taken, leaving aside the NULL ones and the other one is **NO GOSSIPING** which takes only into consideration the answers from data sources that fulfill the constraints or conditions added into the user query, and ignoring all of the inconsistent answers. Another famous strategy based on the metadata stored in the GS is **TRUST YOUR FRIENDS** in which data are preferred to be taken from one data source over another based on the user preference in the query or automatically based on some quality criteria such as Timestamp, Accuracy, Cost,…[2], [9],[10],[11]

This strategy doesn't take extensive computations because it depends on pre-taken preferences and decisions based on either data sources or metadata. On the other hand, it might not give good results in terms of the accuracy and precision because it doesn't take into account all factors in solving the conflicts based on the metadata or the data values in the data sources such as the nature of the values and its frequency, the dependency between attributes.

2.3. CONFLICT RESOLUTION. This strategy examines all the data values and metadata before making any decisions about the way to handle conflicts. Two sub strategies to be considered under conflict resolution are: **deciding strategy** and **mediating strategy**.

a) **Deciding Strategy.** This strategy resolves conflicts using the existing actual values depending on the data values or the metadata[12]. Two common techniques are mostly used under this strategy based on instance values, which are: **CRY WITH THE WOLVES** in which we select the value that appears the most among the conflicting values, while the other technique **ROLE THE DICE** picks a value at random from the conflicting ones. One of the famous techniques based on the metadata is **KEEP UP TO DATE** which uses the timestamp metadata about the data sources, attributes and the data to select from the conflicting values based on recency.

b) **Mediating Strategy.** This strategy chooses a value that is not one of the exiting actual values taking into account the data values and/or the stored metadata. An example of a technique of this strategy is **MEET IN THE MIDDLE** which invents a new value from the conflicting values to represent them by many ways: like taking the mean of the values for example. Other techniques even use higher-level criteria such as provenance to resolve inconsistencies[13].

This strategy is more computationally expensive than the previous two strategies because all of its techniques require computations in the run-time through accessing the actual data and the metadata. Although its computational problem, this strategy might give more accurate answers without any user intervention.

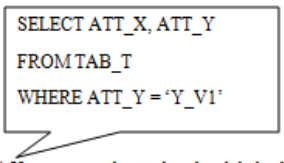
Some systems were developed that used one or more of the above strategies and techniques in handling data conflicts in relational integrated data sources such as: HumMer[14] and ConQuer[15], or extended SQL to be able to work with data fusion[14], [16]. Other techniques were developed to work with probabilistic databases to handle inconsistent data[17], [18][19].

3. ANSWERS FUSION SCORING TECHNIQUE

In this section, we will focus on illustrating our new answers fusion technique. Our proposed technique resolves conflicts automatically based on decisions relying on both the data values in the instances and the metadata stored about each of the contributing data sources and attributes. So, it is a mixed deciding strategy for conflict resolution using both instances data and metadata.

Our fusion technique is based on two basic scores which are: the dependency between all attributes in each table in data source which is a **measure of how well the attribute values are correlated in each data source**, and the other score is the relative frequency for each of the conflicting data values which is a measure of **how popular is the data value in its cluster of duplicated records**.

For simplicity, we will use the simple user query and the couple of data sources shown in Figure 4 (a), (b) respectively. So, as a preprocessing step for all data sources in the data integration environment, the dependency between all attributes in each table in data source is calculated and tored in the Dependency Statistics Module using the *Attributes Dependency Score.*



a) User query submitted to the global schema

DS1.TAB_T			
ATT_A	ATT_B	ATT_X	ATT_Y
A_V1	B_V2	X_V2	Y_V1
A_V2	B_V2	X_V2	Y_V2
A_V1	B_V1	X_V3	Y_V4
A_V1	B_V1	X_V3	Y_V1
A_V3	B_V4	X_V4	Y_V4
A_V3	B_V3	X_V3	Y_V3
A_V4	B_V4	X_V4	Y_V4
A_V2	B_V1	X_V2	Y_V3
A_V1	B_V1	X_V3	Y_V1
A_V2	B_V3	X_V3	Y_V3

DS2.TAB_T			
ATT_A	ATT_C	ATT_X	ATT_Y
A_V1	C_V2	X_V2	Y_V1
A_V2	C_V2	X_V2	Y_V2
A_V1	C_V1	X_V3	Y_V3
A_V1	C_V1	X_V3	Y_V5
A_V3	C_V4	X_V4	Y_V4
A_V3	C_V3	X_V3	Y_V5
A_V4	C_V4	X_V4	Y_V4
A_V2	C_V1	X_V3	Y_V3
A_V1	C_V1	X_V5	Y_V1
A_V2	C_V3	X_V3	Y_V3

B) Data sources contributing in answering the posted query

Figure 4: a) a sample user query submitted to the global schema; b) data sources contributing in answering the query

Definition 3.1 (Attributes Dependency Score)
Attributes Dependency Score is the relative frequency for the summation of all value pairs having support greater than one.
It determines the level of dependency between two attributes based on their values.
The score value ranges from zero to one. The closer the score to one; the highest the dependency between the attributes values.

Equation 3.1 (Attributes Dependency Score)
Let C1 and C2 represents two attributes in the same data source DS, and let D1and D2 be the list of values in C1 and C2 respectively and DS(Card) to present the number of records in the data source. The score of the attributes dependency D (C1, C2) between C1 and C2 is defined as:

$$\text{Score } D(C1, C2) = \frac{\sum_{k=0}^{DS(Card)} \text{Count } (D1(k), D2(k))}{DS(Card)}$$

Where Count (D1 (k), D2 (k)) > 1.

A sample of how the *attributes dependency score* was calculated between ATT_A and ATT_X in DS1.TAB_T is shown in Table 1. We consider the *attributes dependency score* between the semantically correlated attributes given by the attributes semantic rules to equal 1 without any further calculations. Given all of the attributes dependency scores and the query fragments, we will calculate the local detectors. For each of the query requested attributes, the local detectors within each data source consists of the lowest dependent attribute in addition to the intersection of attributes between all of the contributing dependency lists having dependency score greater than ZERO. Of course, we exclude the query attributes from the selection to ensure better duplicate detection. In our example, the set of local detectors are: {ATT_A, ATT_B} and {ATT_A, ATT_C} for DS1.TAB_T and DS2.TAB_T respectively.

DS1.TAB_T		
ATT_A	ATT_X	Frequency
A_V1	X_V2	1
A_V2	X_V2	2
A_V1	X_V3	3
A_V3	X_V4	1
A_V3	X_V3	1
A_V4	X_V4	1
A_V2	X_V3	1
Score D(ATT_A, ATT_X)		0.5

Table 1: *Attributes Dependency Score* between attributes ATT_A and ATT_X in data source DS1.TAB_T

Then, we define the unified detectors set, over all contributing data sources, to be the intersection of all the sets of local detectors. In our case, the unified detector set will be {ATT_A} which will be used to calculate the Local Attributes Scoring for the requested attributes of the query which are ATT_X and ATT_Y for each data source.

Definition 3.2 (Local Attributes Scoring "LAS")
Local Attributes Scoring (LAS) is the average of the dependency scores between the attribute and all unified detectors; multiplied by the complement of the summation of the dependency scores between all unified detectors.
It indicates the level of attribute dependency on the unified detectors.
The score value ranges from zero to one. The closer the score to one; the lowest the dependency between the attribute and the unified detectors and the highest the dependency between the unified detectors.

Equation 3.2 (Local Attributes Scoring "LAS")
Let UN_DET be the list of unified detectors and UN_DET (Card) to present the number of unified detectors over all contributing data sources. We define the LAS for a given attribute ATT to be:

$$\text{LAS (ATT)} = \frac{\sum_{k=0}^{UN\_DET(Card)} \text{Score D(ATT, UN\_DET(k))}}{UN\_DET(Card)} * \left(1 - \frac{\sum_{k=0,j=1}^{\text{UN\_DET(Card)}} \text{Score D(UN\_DET(k), UN\_DET(j))}}{C_2^{UN\_DET(Card)}}\right)$$

Where $k \neq j$.

Thus, we get the complement of the summation of attributes dependency scores between detectors, to make sure that we minimize the transitive dependency between detectors. We calculate the LAS for all contributing attributes for all data sources, and integrate them into one set of answers as in Table 2 (a) and (b) respectively.

DS1.TAB_T				
ATT_A	ATT_X	LAS	ATT_Y	LAS
A_V1	X_V2	0.5	Y_V1	0.5
A_V1	X_V3	0.5	Y_V1	0.5
A_V1	X_V1	0.5	Y_V1	0.5

DS2.TAB_T				
ATT_A	ATT_X	LAS	ATT_Y	LAS
A_V1	X_V2	0.4	Y_V1	0.4
A_V1	X_V5	0.4	Y_V1	0.4

a) LAS for all attributes of the fragmented answers from data sources

Integrated Answers					
ATT_A	ATT_X	LAS	ATT_Y	LAS	DS
A_V1	X_V2	0.5	Y_V1	0.5	DS1.TAB_T
A_V1	X_V3	0.5	Y_V1	0.5	DS1.TAB_T
A_V1	X_V1	0.5	Y_V1	0.5	DS1.TAB_T
A_V1	X_V2	0.4	Y_V1	0.4	DS2.TAB_T
A_V1	X_V5	0.4	Y_V1	0.4	DS2.TAB_T

b) Integrated answers with LAS scores and data source for each answer

Table 2: a) LAS for all requested attributes for all data sources; b) Query answers after integration with LAS and data source for each answer.

Many token-based techniques have been used in the past to detect duplicates from different data sources and cluster them into groups. We used the enhanced token-based technique for this job, and provide it with the set of unified detectors for better detection of duplicates. As shown in Table 3 (a) we got three clusters; one of them with only one record (shaded in grey). So, the two clusters named C1 and C3 will be the only clusters that will be passed to the data fusion process to calculate GAS for each of its their conflicting attribute values.

Furthermore, we can remove the unified detectors and the data source from the answers because they were only used for the duplicate detection and clustering. The unified detectors are used for enhancing the detection and clustering of duplicates, and the data source to ensure the fact that we don't get duplicates from the same data source as we are assuming that we have clean and consistent data sources with no duplicates in them.

Definition 3.3 (Global Attributes Scoring "GAS")
Global Attributes Scoring (GAS) is the cluster relative frequency for the attribute value.
It determines the popularity level of the attribute value within its cluster values.
The score value ranges from zero to one. The closer the score to one; the highest the popularity and the frequency of the value within its cluster.

Equation 3.3 (Global Attributes Scoring "GAS")
Let ATT be the attribute for which we want to calculate GAS, and let D represents the list of values in ATT respectively and CLUS (Card) to present the number of records in the cluster. We define the GAS for a given attribute ATT in a given record to be:

$$\text{GAS (ATT)} = \frac{\sum_{k=0}^{\text{CLUS(Card)}} \text{Count(D(k))}}{\text{CLUS(Card)}}$$

In other words, we calculate the relative frequency in the cluster for each value and assign it to its attribute to be its GAS.

Within each cluster, we work with each attribute separately to calculate its *Total Attributes Scoring* (TAS) in order to apply the fusion rules.

Definition 3.4 (Total Attributes Scoring "TAS")
Total Attributes Scoring (TAS) is the average between both the LAS and GAS values for this attribute.
It is a mixed score indicating both the level of popularity of the attribute value within its cluster and the level of attribute dependency on the unified detectors.
The score value ranges from zero to one. The closer the score to one; the highest the frequency of the value within its cluster and the lowest the dependency on the unified detectors.

Equation 3.4 (Total Attributes Scoring "TAS")
Let ATT be the attribute for which we want to calculate TAS, and let LAS and GAS represents its *Local Attributes Scoring* and *Global Attributes Scoring*. We define the TAS for a given attribute ATT in a given record to be:

$$\text{TAS (ATT)} = \frac{LAS\ (ATT) + GAS\ (ATT)}{2}$$

Where GAS (ATT) ≠ 1; and TAS (ATT) = 1 otherwise.

As in Table 3 (b) two conflicting values appear to have the same GAS (0.5). So, we apply our fusion rules to take the value with the highest LAS or the highest TAS which will be the same. In case the LAS value in coincidence is the same for both conflicting values, we can recalculate the LAS based on local detectors instead of the unified detectors, and still choose the value with the highest LAS.

Finally, we use these TAS values to calculate the answers fusion score that will be shown to the user.

Clustered Answers						
Cluster #	ATT_A	ATT_X	LAS	ATT_Y	LAS	DS
C1	A_V1	X_V2	0.5	Y_V1	0.5	DS1.TAB_T
	A_V1	X_V2	0.4	Y_V1	0.4	DS2.TAB_T
C2	A_V1	X_V1	0.5	Y_V1	0.5	DS1.TAB_T
C3	A_V1	X_V3	0.5	Y_V1	0.5	DS1.TAB_T
	A_V1	X_V5	0.4	Y_V1	0.4	DS2.TAB_T

a) Clustered duplicates with LAS scores and data source for each answer

	Clustered Answers with GAS and applying Data Fusion							
Cluster #	ATT_X	LAS	GAS	TAS	ATT_Y	LAS	GAS	TAS
C1	X_V2	0.5	1	1	Y_V1	0.5	1	1
	X_V2	0.4	1	1	Y_V1	0.4	1	1
C3	X_V3	0.5	0.5	0.5	Y_V1	0.5	1	1
	X_V5	0.4	0.5	0.45	Y_V1	0.4	1	1

b) LAS, GAS, TAS for each attribute within the clusters with data conflicts (grey color)

Table 3: a) Clustered duplicates with LAS scores and data source for each answer; b) LAS, GAS, TAS for each attribute within the clusters with data conflicts (grey color).

Definition 3.5 (Answers Fusion Scoring "AFS")
Answers Fusion Scoring (AFS) is the percentage of average of all attributes TAS values of this answer.
It determines the credibility level of the fused answer.
The score value ranges from zero to 100%. The closer the score to 100%; the highest the credibility to the fused answer.

Equation 3.5 (Answers Fusion Scoring "AFS")
Let ANS be the fused answer for which we want to calculate AFS, and let TAS_LST represents the list of TAS for all attributes in this answer, and ATT (Card) to indicates the number of attributes in the answer . We define the AFS for the given fused answer to be:

$$\text{AFS (ANS)} = \frac{\sum_{k=0}^{\text{ATT(Card)}} \text{TAS (TAS\_LST(k))}}{\text{ATT(Card)}} * 100$$

Consequently, the final answers presented in Table 4 with the corresponding AFS gives the user a certainty percentage about the answer.

Final Answers with AFS		
ATT_X	**ATT_Y**	**AFS**
X_V1	Y_V1	100%
X_V2	Y_V1	100%
X_V3	Y_V1	75%

Final answers with AFS certainty percentage

Table 4: Final answers with AFS certainty percentage

Using this technique, we return answers with AFS equals 100% if all of the duplicates have the same conflicting data values. Otherwise, our technique favors data values based on a combination of its higher relative frequency in the cluster, and its lower dependency on the unified detectors over all contributing data sources.

As most data in real world, especially those stored in databases, contain data and attributes having dome kind of dependency and correlation between each other; our technique uses this fact to resolve conflicts between duplicated records. So, it will work well if used in such environments, but might face some difficulties if worked with unstructured or semi-structured data sources. Further techniques should be used to detect semantic rules within such data sources in integration with our proposed scoring methods.

4. FIVE STEPS DATA FUSION FRAMEWORK

We integrated all of the data fusion steps mentioned in the previous section into a data fusion framework. We modified and extended the common three steps data fusion process shown in Figure 2, which goes through schema mapping, duplicate detection and data fusion, into a new Five Steps Data Fusion Framework as shown in Figure 5.

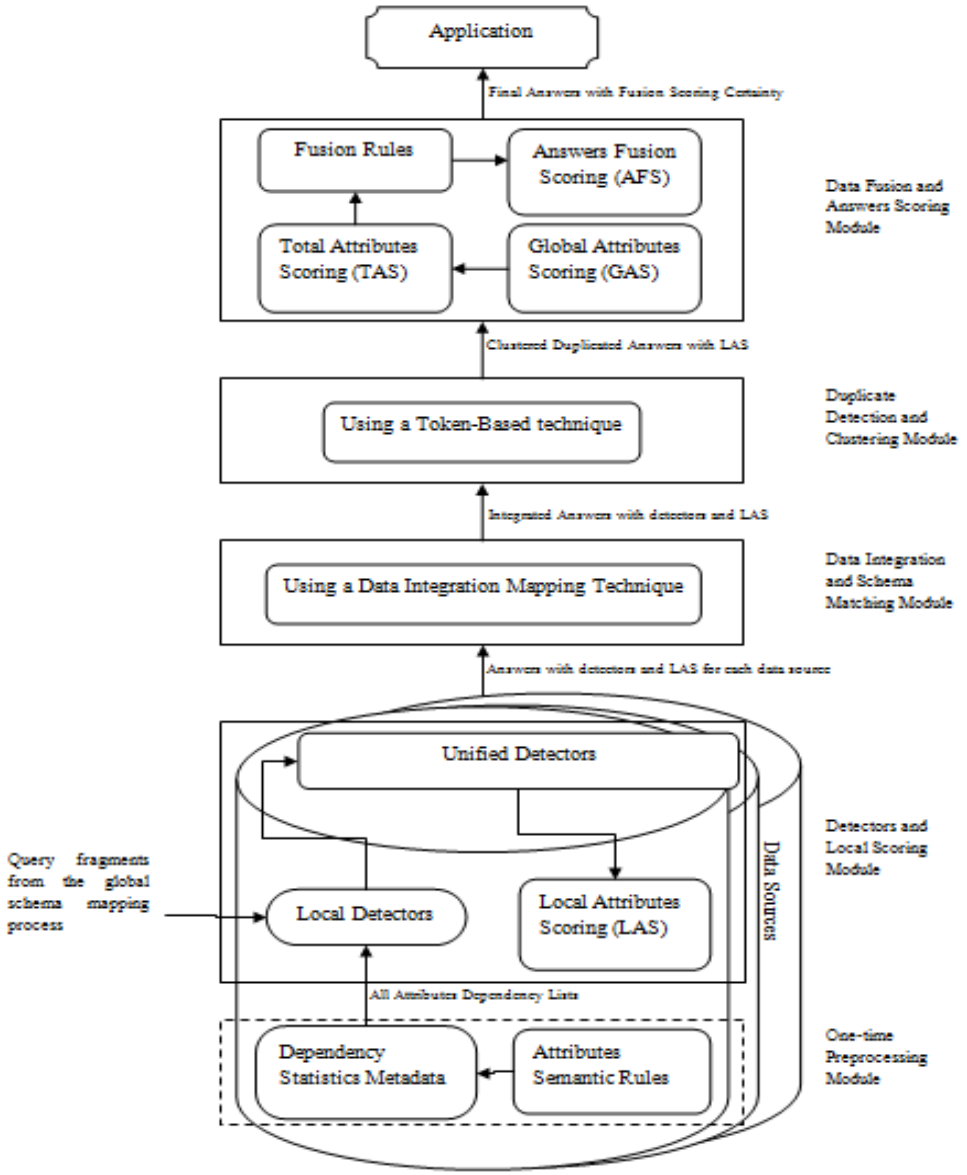


Figure 5: Five Steps Data Fusion Framework.

Our data fusion framework consists of five modules: two added new modules which are the **one-time processing module** and the **detectors and local scoring module**, in addition to some modifications to the common data fusion module with the two other common modules (Schema Matching, and Duplicate Detection) remain unchanged.

We explain in details each of the five modules in the order of execution.

a) One-time Preprocessing Module.

This is one of the two new modules. It is executed once per data sources update. The idea behind this module is to reduce the computational time in case it was executed in the run-time.

It will work efficiently if our data sources don't have extensive lifetime updates, which doesn't include data sources for banks, insurance, stocks and other fast-updated data sources. This module is executed on each data source separately, and it consists of two components: **Attributes Semantic Rules,** and **Dependency Statistics Metadata.** The firstly mentioned component stores the pre-defined semantic dependency rules about some attributes within the data source. One famous example is the semantic rule between the address and zip code. These semantic rules are usually defined by the data source designer, and should be considered without further processing by the other component. The Dependency Statistics Metadata component is the most important component in this module, as it has all of the computations. It calculates the dependency between all of the attributes within each table of the data source. Then for each attribute, the list of association scores with all attributes is sorted in ascending order and passed to the Local Detectors Component in the Detectors and Local Scoring Module.

b) Detectors and Local Scoring Module.

This is a novel module as well, and is considered the real-time start of the data fusion five steps process. As most of the computations were done in the previous module, this module applies some metadata rules to choose detectors, and less computation to calculate the LAS for each attribute requested in the user query. Three components within this module: **Local Detectors, Unified Detectors and Local Attributes Scoring (LAS)**. After the user submits the query to the global schema, the mapping process parse and fragment the posted query and send the query fragments to the contributing data sources. The Local Detectors component within each data source gets the query fragments and all attributes dependency lists from the first module, and applies the local detectors rule on the user's requested attributes only. Then, the next component Unified Detectors runs over all of the data sources local detectors, and takes the intersection between all lists to be the unified detectors for all contributing data sources. If no intersection appears, then all of the fragment answers will be presented to the user for selection. The unified detectors are posted back to the contributed data sources to calculate the LAS for each attribute. Finally, the query answers, including the detectors and LAS calculated for each attribute, are posted to the next module for integration.

c) Data Integration and Schema Matching Module.

This module was mentioned in the literature, which is used to match the data sources schemas and do data integration. We assumed the all of the conflicts in the schema matching were resolved and we only do data integration to all of the answers coming from the previous module. So, this module only union all of fragmented answers into an integrated set of answers and passes it to the next module.

d) Duplicate Detection and Clustering Module.

Many systems and applications applied this module to detect possible duplicates representing the same object[20]. Some techniques uses the theory of record linkage[21]; others rely on similarity measures[22]; while other people use tokens instead of the whole record to detect duplicates[23], [24]. Some other work was concerned in combining records from probabilistic databases[25], and other to find duplicated in XML data models[26].

Using **an enhanced smart token-based technique**[27], we claim that using the detectors attached to the integrated answers could enhance its detection of duplicates. Furthermore, the detected duplicated will be clustered into groups with a cluster number assigned to each group. Non-clustered answers won't pass by the data fusion module and will be presented to the user

among the final result with a percentage of fusion certainty score equal to 100%. We don't need the detectors anymore after this module because we just use them for the detection and clustering of duplicates. So, we can reduce the size of the answers by removing the unified detectors attributes. As for the clustered answers, containing duplicated answers with LAS assigned to attributes, they will be passed to the next and last module for applying data fusion.

e) Data Fusion and Answers Scoring Module.

This module was modified to include four sequential components: **Global Attributes Scoring (GAS)**, **Total Attributes Scoring (TAS)**, **Fusion Rules**, and **Answers Fusion Scoring (AFS)**. These four components are applied to each one of the clusters coming from the former module. The Global Attributes Scoring component is calculated based on the relative frequency of each answer's data value along the cluster. In case those data values of the attribute are different, we calculate the TAS as the mean value of the corresponding LAS and GAS. The fusion rules are applied to resolve the conflicting attribute values within the cluster, by simply selecting the value with the highest TAS. After resolving all conflicting data values, the Answers Fusion Scoring component calculates the total score for each of the fused answers by taking the average TAS of all attributes in the fused answer. Finally, this module returns the final answers with fusion scoring certainty assigned to each one of these answers.

5. CONCLUSION AND FUTURE WORK

In this paper, we considered a new way to resolve data inconsistencies through data fusion based on some statistical scores. These scores are calculated based on the pair-wise dependency scores between the query requested attributes and the duplicate detectors on one side, and the relative frequency of the data values within its cluster from the other side. We also used the proposed scoring technique to define the local detectors for each of the data sources contributing in answering the user query, towards defining the set of unified detector over all data sources. These contributions were developed and integrated into a new five steps data fusion framework which extended the common framework used in literature.

This technique could be applied on any matching application to give you a fused answer with certainty of matching such as social network applications, or face detection applications if provided with the suitable data values and appropriate attributes semantic rules. It could be extended to help improving the work done in natural language processing, suggesting phrase and paragraph structures.

Another open problem to be considered is how we can determine the duplicate detectors and fusion detectors from the submitted user query using some preprocessing metadata, and if these two types of detectors should be the same, or differ based on the submitted query. One last idea is if we can substitute the scoring methods used to be based on information gain instead of dependency and relative frequency score.

REFERENCES

[1] L. Xu and D. W. Embley, "Combining the Best of Global-as-View and Local-as-View for Data Integration," in *In Proc. of the 3rd International Conference ISTA*, 2004, pp. 123–135.

[2] A. Z. EL Qutaany, A. H. El Bastawissy, and O. Hegazy, "A Technique for Mutual Inconsistencies Detection and Resolution in Virtual Data Integration Environment," in *Informatics and Systems (INFOS), 2010 The 7th International Conference on. IEEE*, 2010, pp. 1–8.

[3] F. Naumann and J. Bleiholder, "Data Fusion in Three Steps : Resolving Inconsistencies at Schema- , Tuple- , and Value-level," *IEEE Data Eng.*, vol. 29, no. 2, pp. 21–31, 2006.

[4] L. Bertossi, J. Chomicki, and C. Guti, "Consistent Answers from Integrated Data Sources," *In Flexible Query Answering Systems*, pp. 71–85, 2002.

[5] X. L. Dong, A. Halevy, and C. Yu, "Data integration with uncertainty," *The VLDB Journal*, vol. 18, no. 2, pp. 469–500, Nov. 2008.

[6] J. Bleiholder and F. Naumann, *Conflict Handling Strategies in an Integrated Information System*. 2006.

[7] J. Bleiholder and F. Naumann, "Data fusion," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–41, Dec. 2008.

[8] X. Dong and F. Naumann, "Data Fusion – Resolving Data Conflicts for Integration," in *Proceedings of the VLDB Endowment 2(2)*, 2009, pp. 1645–1655.

[9] A. Motro and P. Anokhin, "Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources," *Information Fusion*, vol. 7, no. 2, pp. 176–196, Jun. 2006.

[10] G. De Giacomo, R. La, V. Salaria, and I.- Roma, "Tackling Inconsistencies in Data Integration through Source Preferences," in *In Proceedings of the 2004 international workshop on Information quality in information systems. ACM*, 2004, pp. 27–34.

[11] X. Wang, H. LIN-PENG, X.-H. XU, Y. ZHANG, and J.-Q. CHEEN, "A Solution for Data Inconsistency in Data Integration," *Journal of information science and engineering*, vol. 27, pp. 681–695, 2011.

[12] P. Anokhin, S. Engineering, and A. Motro, "Use of Meta-data for Value-level Inconsistency Detection and Resolution During Data Integration," *ech. Rep. ISETR-03-06. Department of Information and Software Engineering. George Mason University, USA.*, 2003.

[13] Y. Katsis, A. Deutsch, Y. Papakonstantinou, and V. Vassalos, "Inconsistency Resolution in Online Databases," in *In Data Engineering (ICDE), 2010 IEEE 26th International Conference on IEEE*, 2010, pp. 1205–1208.

[14] A. Bilke, J. Bleiholder, F. Naumann, B. Christoph, and M. Weis, "Automatic Data Fusion with HumMer," in *In Proceedings of the 31st international conference on Very large data bases. VLDB Endowment.*, 2005, pp. 1251–1254.

[15] A. Fuxman and E. Fazli, "ConQuer : Efficient Management of Inconsistent Databases," in *In Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM*, 2005, pp. 155–166.

[16] J. Bleiholder and F. Naumann, "Declarative Data Fusion – Syntax , Semantics , and Implementation," *In Advances in Databases and Information Systems*, pp. 58–73, 2005.

[17] S. Maitra and R. Wason, "A Naive Approach for Handling Uncertainty Inherent in Query Optimization of Probabilistic Databases Institute of Information Technology and Management," 2011.

[18] X. Lian, L. Chen, and S. Song, "Consistent query answers in inconsistent probabilistic databases," *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*, vol. 8, p. 303, 2010.

[19] P. Andritsos, a. Fuxman, and R. J. Miller, "Clean Answers over Dirty Databases: A Probabilistic Approach," *22nd International Conference on Data Engineering (ICDE'06)*, pp. 30–30, 2006.

[20] A. K. Elmagarmid and S. Member, "Duplicate Record Detection : A Survey," *Knowledge and Data Engineering, IEEE Transactions*, vol. 19, no. 1, pp. 1–16, 2007.

[21] A. B. Sunteb and I. P. Fellegi, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[22] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, p. 39, 2003.

[23] C. I. Ezeife and T. E. Ohanekwu, "Use of Smart Tokens in Cleaning Integrated Watehouse Data," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 1, no. 2, pp. 1–22, 2005.

[24] T. E. Ohanekwu and C. I. Ezeife, "A Token-Based Data Cleaning Technique for Data Warehouse Systems," in *In Proceedings of the International Workshop on Data Quality in Cooperative Information Systems*, 2003, pp. 21–26.

[25] F. Panse and N. Ritter, "Tuple Merging in Probabilistic Databases," in *In Proceedings of the fourth International Workshop on Management of Uncertain Data (MUD),Singapur*, 2010, pp. 113–127.

[26] M. Weis and F. Naumann, "DogmatiX Tracks down Duplicates in XML," in *In Proceedings of the 2005 ACM SIGMOD international conference on Management of data,ACM*, 2005, pp. 431–442.

[27] A. S. El Zeiny, A. H. El Bastawissy, A. S. Tolba, and O. Hegazy, "An Enhanced Smart Tokens Algorithm for Data Cleansing in Data Warehouses," in *In the Proceedings of the Fifth International Conference on Informatics and Systems (INFOS 2007), Cairo, Egypt*, 2007.