Using Information Gain in Data Fusion and Ranking

[1]MOHAMED M. HAFEZ, [2]ALI H. EL-BASTAWISSY, [1]OSMAN M. HEGAZY
[1]Faculty of Computers and Information, [2]Faculty of Computer Science
[1]Cairo University, [2]University of Modern Sciences and Arts
EGYPT
[1]mhafez@fci-cu.edu.eg, [2]aelbastawissy@msa.eun.eg, [1]o.hegazy@fci-cu.edu.eg

Abstract: Entropy and information gain have been traditionally used to measure association between inputs and outputs. In this paper, Information gain is used to measure and decide the level of dependency or relevance between attributes. A data fusion technique based on information gain measures in a virtual data integration environment is introduced. After the detection and clustering of duplicates, the fused records are ranked and provided to the user in the final answer set with a preference score associated with each answer.

Key-Words: Data integration, data fusion, information gain, duplicates detectors, conflict resolution, ranking answers

1 Introduction

In the virtual data integration environment, the user submits the query to the global unified schema with data stored in local data sources. The Global-as-View (GaV) is a mapping technique used to determine the local data sources contributing in the answer of the user query, and integrate the partial answers of these local data sources. Then, the integrated answers are grouped into clusters of duplicates that are finally passed to the data fusion step to resolve any data conflicts before posting the final answer to the user.

Through the previous steps, many challenges should be faced and resolved in order to give the user a set of clean and consistent answers. The most important challenge to address in this context is inconsistencies. They arise through the different steps in the virtual data integration framework; they could be schema matching inconsistencies, representation inconsistencies, or data inconsistencies. In our technique, we assume that both schema matching and representation inconsistencies are handled; thus we focus only on resolving data inconsistencies.

In literature, many trials were done to handle data inconsistencies; some favored conflict ignorance, others preferred the avoidance of conflicts, and the rest used some conflict resolution techniques. In this paper, we introduce a new conflict resolution technique which provides a way to favor values over others based on some preprocessing and real time calculations over the conflicting attributes. Many possible obstacles should be addressed in both duplicate detection and data fusion steps. One of the main challenges to consider in the duplicate detection step, is how to improve the findings of duplicates, is it through some predefined metadata or through traditional ways like record linkage for example? Another important problem to arise after integration and duplicates detection is that within each cluster of duplicates, some values might conflict between different records representing the same object. In order to resolve such conflicts, should we choose from the conflicting values based on metadata or the values themselves or should we invent new data values from the conflicting ones based on some criteria? Therefore, the challenge to be addressed here is to find the most appropriate technique to resolve conflicts and complete the data fusion process and provide the user with suitable answers.

This paper is organized as follows; in section 2 we classify and explain the related work. In section 3, the proposed technique for data fusion based on information gain is explained in details. Then, the conclusion and future work are presented in section 4.

2 Related Work

We are working using a four steps data fusion framework shown in Figure 1, which begins with a pre-processing step and ends with the data fusion step, rather than the frameworks in [1][2].

The preprocessing idea was presented originally in [2] for computational purposes, as to save time by calculating attributes dependency in

ISBN: 978-960-474-361-2

each data source offline before identifying the local and unified detectors later after the user submits the query. We use the same idea of the preprocessing step to calculate some gain scores which are used also to define the detectors used in the duplicate detection and data fusion steps.

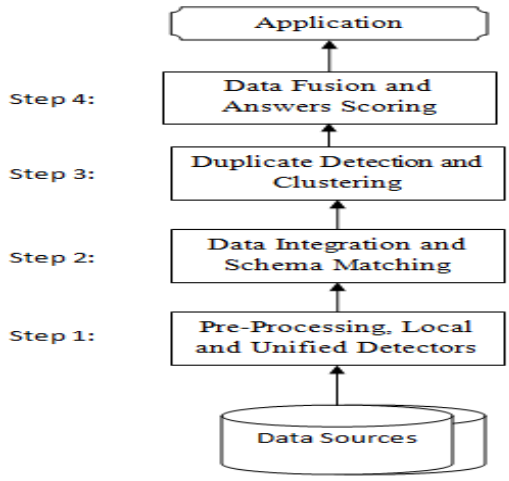


Figure 1: Four Steps of Data Fusion

Partial answers returned from the local data sources need to be integrated without any inconsistencies. May techniques were mentioned to do the mapping and integration between the contributing data sources such as GaV, LaV and GLaV [3]. We assume that all of the schema matching inconsistencies are resolved and the contributing data sources requested attributes are mapped together and their partial query answers are integrated through using the GaV approach.

The problem of detecting duplicated records has been addressed by researches working on various types of data sources in different contexts and environments [4]. One possible classification for such techniques is according to the type of the technique or algorithm; as some techniques follow the common record linkage approach [5][6]. Others used string matching and similarity techniques [7]. Another common classification is based on the type of data sources they are using; much work has been done on relational data [8], while other work uses XML data models to find such duplicates and group them into clusters [9][10][11][12].

Through the process of working with the integrated answers to detect duplicates; some approaches use the whole record to measure the similarity between records, while others use tokens or keywords instead to detect duplicates [13][14]. We will use one of the techniques that uses tokens to detect duplicates which is an enhanced **smart token-based technique**[15]. We will provide the technique with the integrated answers including the requested query attributes in addition to the unified detectors in order to improve the duplicate detection and clustering process.

Many strategies were developed to handle data conflicts in the data fusion step, some of them were repeatedly mentioned in the literature. These conflict handling strategies, shown in Figure 2, can be classified into three main classes based on the way of handling conflicting data: ignorance, avoidance, and resolutions [2][16][17][18].

Conflict Ignorance. In this class, data conflicts are left to the user to handle them. Two famous techniques are used under this class:

PASS IT ON: Takes all conflicting values and passes them to the user to decide.

CONSIDER ALL POSSIBILITIES: Given the set of conflicting records, it generates all possible combinations of data values as possible query answers, and show them to the user to choose.

Despite the inefficiency and high computational workload of the conflict ignorance strategy, its techniques could be implemented with minimal effort. However, it is ineffective to involve the user in deciding on the correct values because of the lack of the user awareness of both the integration schema and the features of the data sources.

Conflict Avoidance. Decisions are pre-made regarding the data values to be chosen in the data fusion step before returning the final result to the user. The famous techniques within this class are:

TAKE THE INFORMATION: This technique uses the data values stored in the data sources, and it works by taking the non-NULL values from the data sources.

NO GOSSIPING: This technique also uses the data values stored in the data sources, and it takes only into consideration the data sources' answers that satisfy the user query, and ignore all of the inconsistent answers.

TRUST YOUR FRIENDS: This technique uses some stored metadata instead of the actual data in the data sources, as data are preferred to be taken from a specific data source based on some user preferences or quality criteria, to be added manually or automatically to the submitted user query, such as Timestamp, Accuracy, Cost,…[19] [20][21].

Because most decisions are pre-made either based on metadata or the data values stored in the

data sources, conflict avoidance techniques doesn't take extensive runtime computations. However, it might not give precise results to the user due to the exclusion of some important factors such as the level of data and attributes correlation and dependency.

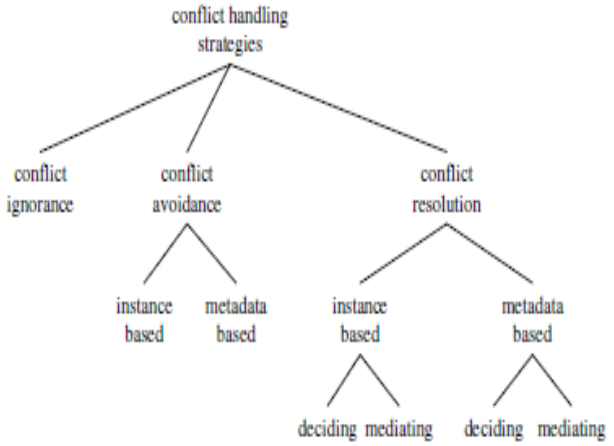


Figure 2: Classification of Conflict Handling Strategies [16]

Conflict Resolution. In this class, data conflicts are resolved without any ignorance or avoidance, as all data values and metadata are examined before making any decisions. Many techniques are mentioned in literature, the famous ones are:

CRY WITH THE WOLVES: This technique just chooses the data value that appears the most among the conflicting values.

ROLE THE DICE: This technique selects a random value from the conflicting data values.

KEEP UP TO DATE: This technique chooses the most recent data value. This is done using the stored timestamp about the data sources, attributes and the data stored in them[22].

Some recent work were developed that resolves conflicts relying on both the data values in the instances and the metadata stored about each of the contributing data sources and attributes [2]. This technique takes into account both the dependency between all attributes in each table which is a measure of attributes dependency, and the relative frequency for each of the conflicting data values which is a measure of popularity of the data value which is the same as the work done using the **CRY WITH THE WOLVES** technique.

MEET IN THE MIDDLE: This technique is different than the previously mentioned techniques as it invents a new value from the conflicting values rather than selecting one of them. This value might take various shapes such as taking the mean of the conflicting values to represent them.

It is obvious that the conflict resolution ends up as the highest computational one among the conflict handling strategies as it examines all the data values and the metadata in run-time. However, the techniques under this class give the users more accurate results without any supervision from the user.

Some systems were developed which considered one or more of the above conflict handling strategies such as: HumMer[23] and ConQuer[24] while others were developed to work with probabilistic databases to handle inconsistent data [25][26][27].

3 Data Fusion Using Information Gain

"Information Gain (IG) is mostly used as an attribute selection measure, and mainly used in conjunction with decision trees"[28]. The attribute with the highest information gain is chosen as the splitting attribute for a set of records. "This attribute minimizes the information needed to classify the records and reflects the least impurity in the resulting partitions"[28][29].

In Literature, information gain was used originally in signals processing mainly in compressing data for transmission or storage purposes. Some applications of information gain in such context include lossless and lossy data compressions, source coding and channel coding[30].

As our previous work in[2] can work properly when the attributes within each data source have high data dependency, it is not be so effective when we work with data sources having low dependency. So, we were motivated to find a solution that fits with the low data dependency problem, and still be valid for high data dependency. Therefore, it was the emergence of the idea of using the concepts of entropy and information gain in this context to measure such dependency and use it to resolve data conflicts in the data fusion step.

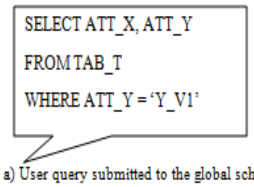
In this section, we introduce the concept of information gain in the field of data integration and data fusion. We focus on presenting how we used information gain in our data fusion and ranking

technique. Our technique resolves data conflicts based on two factors: **the original data values in the data sources on one hand**, and **some stored metadata and IG scores about the contributing data sources and their attributes**.

In order to explain the proposed technique in details, we use the submitted user query and the contributing data sources shown in Figure 3 (a), (b) respectively.

As a pre-processing step for each data source in the existing data integration environment, we calculate **the information gain for each attribute and the gain for the other attributes if used to partition this attribute**. This is the same procedure we follow when using information gain in the decision trees context.

So, in order to calculate the required information gain, three steps should be followed: **First, we calculate the entropy for the target attribute** which is calculated using equation 1 indicating how the values are distributed through this attribute. **Second, we calculate the expected information needed to classify a record in the target attribute based on the partitioning of the other attributes** as in equation 2. **Third, the information gain for each attribute on the target attribute is calculated** using equation 3, indicating how much would be gained by branching the target attribute based on another attribute. We follow the three steps through an example to show how we calculate the information gain from the partitioning of attribute ATT_A by attribute ATT_B in data source DS1.TAB_T.



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 3: a) submitted user query to global schema; b) data sources contributing in answering the submitted query [2]

Definition 1 (Attribute Entropy)
Attribute Entropy is a measure of variation of attribute values.

As it gives a measure of variation in a set of values; the more variation in the values and the number of values classes in the attribute, the higher the entropy of that attribute. The attribute entropy can't be negative, but it will equal to ZERO if the attribute has only one class of values.

Equation 1 (Attribute Entropy)
Let ATT be the attribute for which we want to calculate the entropy of its values, and let C represents the list of ATT classes of values. Let Ci(Card) to present the number of values in calss (i) and ATT(Card) to present the number of values in the given attribute ATT. Pi to present the probability to choose a value of the calss (i) from the list of values in attribute ATT. We define the attribute entropy for the attribute ATT to be:

$$\text{Info (ATT)} = -\sum_{i=1}^{m} \frac{\text{Ci(Card)}}{\text{ATT(Card)}} \text{Pi} * \log_2(\text{Pi})$$

First, we apply the above equation 1 to calculate the *attribute entropy* for ATT_A in DS1.TAB_T, we find that it has 4 values of (A_V1), 3 values of (A_V2), 2 values for (A_V3) and only one equals (A_V4). So, the entropy of ATT_A is calculated as:

Info (ATT_A) = -4/10 $\log_2$ 4/10 -3/10 $\log_2$ 3/10 -2/10 $\log_2$ 2/10-1/10 $\log_2$ 1/10 = **1.8464 bits**

Definition 2 (Expected Information)
The *expected information needed to classify a record in an attribute according to the partitioning of another attribute*, is similar to how we define the conditional probability of an attribute given another one, but consider it for classes of attribute values instead of the whole attribute.

It indicates the level of dependency between the partitioning and the partitioned attributes. It can't be higher than the partitioned attribute entropy and can't be negative. As it indicates dependency; the lower the expected information, the better of the partitioning attribute to classify the records of the partitioned attribute and the more the dependency between the two attributes.

ISBN: 978-960-474-361-2

Equation 2 (Expected Information)
Let ATT be the attribute to be partitioned and P_ATT be the attribute to be used to partition ATT. Let C represents the list of ATT classes of values. Let Ci(Card) to present the number of values in calss (i) and ATT(Card) to present the number of values in ATT. So, we define the expected information needed to partition attribute ATT according to attribute P_ATT to be:

$$\mathrm{Info}_{\mathrm{P\_ATT}}(\mathrm{ATT}) = \sum_{i=1}^{m} \frac{C_i\,(\mathrm{Card})}{\mathrm{ATT}\,(\mathrm{Card})} * \mathrm{Info}(C_i)$$

Second, we apply the above equation 2 to calculate the *expected information* needed to classify a record in ATT_A based on the partitioning of ATT_B in DS1.TAB_T, we find that B_V1 appeared 4 times (3 times with A_V1 and 1 time with A_V2), B_V2 appeared 2 times (1 with A_V1 and 1 with A_V2), and so on... So, the expected information for ATT_A based on ATT_B is:

$\mathrm{Info}_{\mathrm{ATT\_B}}$ (ATT_A) = 4/10 * (-3/4 $\log_2$ 3/4 -1/4 $\log_2$ 1/4) + 2/10 * (-1/2 $\log_2$ 1/2 -1/2 $\log_2$ 1/2) + 2/10 * (-1/2 $\log_2$ 1/2 -1/2 $\log_2$ 1/2) + 2/10 * (-1/2 $\log_2$ 1/2 -1/2 $\log_2$ 1/2) = **0.9245 bits**

Definition 3 (Information Gain for an attribute to partition another)
Information Gain is defined as the difference between the partitioned attribute entropy and the expected information needed to partition this attribute based on another given attribute.
In other words it tells us how much we gained by branching an attribute on another attribute. The highest the gain for the partitioning attribute, the better its nomination to classify the records of the partitioned attribute.

Equation 3 (Information Gain for an attribute to partition another)
Let ATT be the attribute to be partitioned, and P_ATT to be the partitioning attribute. Let Info (ATT) to the entropy of ATT, and $\mathrm{Info}_{\mathrm{P\_ATT}}$ (ATT) to be the expected information needed to partition ATT according to P_ATT. We define the gain of such partitioning to be:

$$\mathrm{Gain}\,(\mathrm{P\_ATT\ on\ ATT}) = \mathrm{Info}\,(\mathrm{ATT}) - \mathrm{Info}_{\mathrm{P\_ATT}}\,(\mathrm{ATT})$$

Third, we apply the above equation 3 to calculate the information gain from the partitioning of ATT_A by ATT_B in DS1.TAB_T, which is simply the difference between ATT_A entropy and the expected information for ATT_A based on ATT_B. So, the information gained by branching ATT_A based on ATT_B is calculated as:

Gain (ATT_B on ATT_A) = Info (ATT_A) - $\mathrm{Info}_{\mathrm{ATT\_B}}$ (ATT_A) = 1.8464 - 0.9245 = **0.9219 bits**

Similarly if we follow the same equations, we can find that:
Gain (ATT_X on ATT_A) =0.6855 bits and **Gain (ATT_Y on ATT_A) =1.0955 bits**

Summarizing what we are trying to do, we need to find the information gain for each attribute if partitioned by another attribute. **A good question to be addressed is why we are doing all of this?** A quick answer is to find the local detectors for each data source. The local detectors of a data source are those attributes if their values were added to any returned query answer, it will be the least likely to have incorrect duplicates (records that will be marked incorrectly as duplicates). Therefore, to find such local detectors we calculate information gains. As the calculated information gain indicates how much an attribute can be selected as a splitter to another attribute. Thus, it indicates how much the two attributes (the splitter and the splitted) are correlated or in other words how much is the splitted attribute dependent on the splitter attribute. To conclude, **the lowest the information gain, the better the splitter to be selected as a detector for the splitted attribute.**

So, we sort the detectors for each of the query requested attributes according to the information gains in ascending order. Therefore, the detectors for ATT_A in DS1.TAB_T are {ATT_X, ATT_B, ATT_Y}.

We define the **local detectors for the whole data source** to be the lowest attribute information gain for each of the query requested attributes in addition to the intersection of the attributes between all of the contributed information gain lists having gain not equal to the basic entropy of the requested attribute. In our case, the set of local detectors are: {ATT_A, ATT_B} and {ATT_A, ATT_C} for both DS1.TAB_T and DS2.TAB_T respectively.

Then, We define **the unified detectors set over all contributing data sources** to be the intersection of all the sets of local detectors [2]. In our case, the unified detectors are {ATT_A}.

Obviously, the pre-processing step is to get the unified detectors for all of the contributing data sources. Part of this step could be done offline for

ISBN: 978-960-474-361-2

each attribute in each data source to sort their detectors. Then, after the submitting the query; the procedure to get the local and unified detectors should be narrowed to only those attributes participating in answering the user query.

As for our example, given the user query and {ATT_A}; we can assume the integration is done between all of the local answers through the unified global schema using the GaV technique.

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

Table 1: Clustered duplicated query answers with the data source for each answer

Many techniques were mentioned in literature that were used to detect and mark records as duplicates and group them into clusters. Regarding this issue, we used the **enhanced token-based technique** [18] that we only improved its work by adding the unified detector values to the returned records from each data source. As shown in Table 1 we have three clusters: C1, C2 and C3. For C1 and C2, we don't have conflicts because the *Global Attributes Scoring (GAS)* = 1 for all attributes. GAS is defined as "the cluster relative frequency for the attribute value" [1]. As for C3, we do have to solve one data conflict colored in red in the above table which is either to choose the value for X_V3 or X_V5 for ATT_X.

So, our data fusion problem is to select an attribute data value from the clustered duplicates and fuse it into a single record representing the real world object [1]. In our case, the real world object with ATT_A value equal to A_V1 and ATT_Y value equal to Y_V1. In order to resolve this data conflict, we consider the above Table 1 as a unified data source, and use the non-conflicting records to resolve this conflict. Therefore, we need to know which case is most likely to happen:

*** ATT_X = X_V3 given ATT_A = A_V1 and ATT_Y = Y_V1**
*** ATT_X = X_V5 given ATT_A = A_V1 and ATT_Y = Y_V1**

Thus, to resolve this conflict we get back again to our contribution of using the information gain in this context. That's why we return to each data source and calculate the gain for ATT_X based on both ATT_A & ATT_Y together named as the dependency *gain scoring (DGS)*:

DS1.TAB_T

$Info_{ATT\_A,\ ATT\_Y}$ (ATT_X) = 3/10 * (-2/3 $\log_2$ 2/3 - 1/3 $\log_2$ 1/3) + 2/10 * (-1/2 $\log_2$ 1/2 -1/2 $\log_2$ 1/2) = **0.4754 bits**

DGS (ATT_X) -> Gain ((ATT_A & ATT_Y) on ATT_X) = Info (ATT_X) - $Info_{ATT\_A,\ ATT\_Y}$ (ATT_X) = **1.01 bits**

DS2.TAB_T

$Info_{ATT\_A,\ ATT\_Y}$ (ATT_X) = 2/10 * (-1/2 $\log_2$ 1/2 - 1/2 $\log_2$ 1/2) = **0.2 bits**

DGS (ATT_X) -> Gain ((ATT_A & ATT_Y) on ATT_X) = Info (ATT_X) - $Info_{ATT\_A,ATT\_Y}$(ATT_X) = **1.5609 bits**

From the above two information gain values, we can decide that we gained more information by using ATT_A & ATT_Y from DS2.TAB_T when judging on ATT_X values than those from DS1.TAB_T. In other words, we can detect the value of ATT_X better in DS1.TAB_T than from DS2.TAB_T. Therefore, we prefer ATT_X values from this source (DS1.TAB_T) if we have the values from ATT_A and ATT_Y.

So, we choose **X_V3** from the above table according to the previous calculations.

As illustrated above, we only consider the integrated and clustered records from Table 1 to resolve our data conflict. Of course, one interesting idea is to consider more records from the data sources to give a better fusion and judgment. We can consider those records with a high attribute GAS values in their data sources.

After passing through all steps and resolving data conflicts in the data fusion step; we present the way we use **to score the final query answer** to the user. We use the concept of **higher scoring**. In other words, for each answer in the final set its score is considered a certain percentage indicating how this answer is better than the one next to it. We have the following scores for each attribute: *Global Attributes Scoring (GAS)* as defined and calculated in [1] refereeing to the relative frequency of the value within its cluster, and *Dependency Gain Scoring (DGS)* refereeing to gain

ISBN: 978-960-474-361-2

of splitting the attribute according to the attributes with no data conflicts.

The clustered answers with GAS and DGS scores are shown in Table 2.The ∆ means the absolute difference between the current value and the next highest/lowest value divided by the next highest/lowest value. For example ∆DGS for in C3 for the record with ATT_X equals X_V3 equals 0.35 because it is the absolute difference between 1.01 & 1.5609 divided by the latter.

Clustered Answers with GAS and Dependency Gain Scoring (DGS)										
Cluster #	ATT_X	GAS	∆ GAS	DGS	∆ DGS	ATT_Y	GAS	∆ GAS	DGS	∆ DGS
C1	X_V2	1	0	0	0	Y_V1	1	0	0	0
	X_V2	1	0	0	0	Y_V1	1	0	0	0
C2	X_V1	1	0	0	0	Y_V1	1	0	0	0
C3	X_V3	0.5	0	1.01	0.35	Y_V1	1	0	0	0
	X_V5	0.5	0	1.5609	NA	Y_V1	1	0	0	0

Table 2: Clustered answers with GAS and DGS scores

Then, we calculate a total score for each attribute separately called *Total Attributes Preference Scoring (TAPS)* using equation 4 which is used to calculate the final answers scoring called *Answers Fusion Preference Scoring (AFPS)* using equation 5 that will be associated with each of the possible query answers to the user. The clustered answers with TAPS scores and the final answers with AFPS scores are shown in Table 3 (a) and (b) respectively.

Definition 4 (Total Attributes Preference Scoring)
Total Attributes Preference Scoring (TAPS) is the average between both the GAS and DGS 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 information gain (dependency) based 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 4 (Total Attributes Preference Scoring)
Let ATT be the attribute for which we want to calculate TAS, and let DGS and GAS represents its Dependency Gain Scoring and Global Attributes Scoring. We define the TAS for a given attribute ATT in a given record to be:

$$\text{TAPS (ATT)} = \frac{\Delta \text{ DGS (ATT)} + \Delta \text{ GAS (ATT)}}{2}$$

Where ∆ DGS (ATT) = −(DGS (ATT) - next higher DGS value)/ next higher DGS value,
∆ GAS (ATT) = (GAS (ATT) - next lowest GAS value)/ GAS (ATT),
GAS (ATT) ≠ 1; and TAPS (ATT) = 0 otherwise.

Definition 5 (Answers Fusion Preference Scoring)
Answers Fusion Preference Scoring (AFPS) is the percentage of average of all attributes TAPS values of this answer.
It determines the credibility level of the fused answer over the next best answer.
The score value ranges from zero to 100%. The closer the score to 100%; the highest the credibility to the fused answer over the next best answer.

Equation 5 (Answers Fusion Preference Scoring)
Let ANS be the fused answer for which we want to calculate AFPS, and let TAPS_LST represents the list of TAPS for all attributes in this answer, and ATT (Card) to indicates the number of attributes in the answer . We define the AFPS for the given fused answer to be:

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

Note that using this technique; we are able to associate the final answers with a preference score that indicates how much the answer is better than the next best answer. All of the preference is done after applying our data fusion by preferring the value based on GAS and DGS scores. One can keep all duplicates without applying any data fusion rules, and just rank the answers or the duplicates using the same way that we applied above.

This technique uses the information gain outside its main usage in two scenarios; initially it is used to get the local and unified detectors and later is used in the data fusion and scoring step to give a

ISBN: 978-960-474-361-2

meaning to the final answers returned to the user. We can apply our method to any data source with any level of correlation or dependency between its attributes as we are using entropy and information gain and not depending on attribute correlation as introduced in [2]. Further applications and suggestions are mentioned in the next section.

Clustered Answers with TAPS										
Cluster #	ATT_X	GAS	Δ GAS	Δ DGS	TAPS	ATT_Y	GAS	Δ GAS	Δ DGS	TAPS
C1	X_V2	1	0	0	0	Y_V1	1	0	0	0
	X_V2	1	0	0	0	Y_V1	1	0	0	0
C2	X_V1	1	0	0	0	Y_V1	1	0	0	0
C3	X_V3	0.5	0	0.35	0425	Y_V1	1	0	0	0
	X_V5	0.5	0	NA	NA	Y_V1	1	0	0	0

Final Answers with AFPS		
ATT_X	ATT_Y	AFPS
X_V1	Y_V1	0%
X_V2	Y_V1	0%
X_V3	Y_V1	21.25%

Table 3: (a) Clustered answer with TAPS scores; (b) Final answers with AFPS scores

4 Conclusion and Future Work

In this paper, we are not only introducing a novel data fusion technique to resolve data conflicts, but also a way of ranking the final fused query answers which was done through using entropy and information gain. The concept of entropy was used in this context to measure the level of variation of values within a given attribute, while the information gain was utilized to measure the level of dependency between attributes. We used the information gain to define the local detectors for each of the data sources contributing in answering the submitted user query, and then it was used in the data fusion step to produce scores which were partially used to associate a preference scoring percentage with each answer in the final set returned to the user. An interesting challenge it to provide a more coherent score than the answers fusion preference scoring (AFPS) that can be used to indicate the accuracy or creditability of the fused answer instead of providing a preference score based on two differently calculated mixed scores.

The provided technique could be utilized to work on any kind of data regardless the level of dependency or correlation. It could be used to fuse and rank possible results returning from a search engine, or to provide preference scores on a possible matching and integration between web account data if we want to work in a web integration environment (emails, bank accounts,…), or to work in social or job networks to suggest on possible friendships or nominate suitable jobs based on matching skills and requirements. Through using the pre-processing step, the technique will provide significant time efficiency for applications with large size of data such as NLP, social network or those built over a cloud computing architecture.

References:

[1] 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.

[2] M. M. Hafez, A. H. El Bastawissy, and O. H. Mohamed, “A Statistical Data Fusion Technique in Virtual Data Integration Environment,” International Journal of Data Mining and Knowledge Management Process, vol. 3, no. 5, pp. 25–38, 2013.

[3] 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.

[4] 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.

[5] 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.

[6] P. Christen and K. Goiser, “Quality and Complexity Measures for Data Linkage and Deduplication,” Quality Measures in Data Mining, pp. 127–151, 2007.

[7] 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.

[8] 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.

[9] 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.

[10] M. Weis, "Fuzzy Duplicate Detection on XML Data," in Proceedings of the 31st VLDB Conference, 2005.

[11] R. M. A. El-ghfar and A. El-bastawissy, "DRTX : A Duplicate Resolution Tool for XML Repositories," International Journal of Computer Science and Network Security, vol. 12, no. 7, pp. 42–50, 2012.

[12] M. Weis and F. Naumann, "Detecting Duplicate Objects in XML Documents.pdf," in International Workshop on Information Quality in Information Systems, 2004.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] S. Maitra and R. Wason, "A Naive Approach for Handling Uncertainty Inherent in Query Optimization of Probabilistic Databases Institute of Information Technology and Management," in Proceedings of the 5th National Conference; INDIACom, 2011.

[26] 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.

[27] 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.

[28] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Second Edi. Elsevier Inc., 2006.

[29] T. Carter, "An introduction to information theory and entropy," Santa Fe, 2007.

[30] E. Liu, E. K. P. Chong, L. L. Scharf, and L. Fellow, "Greedy Adaptive Compression in Signal-Plus-Noise Models," pp. 1–23, 2012.