1-1-2016

An Algorithm for Inferring Big Data Objects Correlation Using Word Net

M. Basel Almourad
Zayed University

Mohammed Hussain
Zayed University

Talal Bonny
University of Sharjah

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 1238 – 1243

Procedia Computer Science

The 3rd International Workshop on Machine Learning and Data Mining for Sensor Networks (MLDM-SN))

An Algorithm for Inferring Big Data Objects Correlation Using Word Net

M. Basel Almourad[a]*, Mohammed Hussain[a], Talal Bonny[b]

[a]*Zayed University, P. O. Box 19282, Dubai, UAE*
[b] *University of Sharjah, P. O. Box 27272, Sharjah, UAE*

Abstract

The value of big data comes from its variety where data is collected from various sources. One of the key big data challenges is identifying which data objects are relevant or refer to the same logical entity across various data sources. This challenge is traditionally known as schema matching. Due to big data velocity traditional approaches to data matching can no longer be used. In this paper we present an approach for inferring data objects correlation. We present our algorithm that relies on the objects meta-data and it consults the Word Net thesaurus

Peer-review under responsibility of the Conference Program Chairs

Keywords: Big data; Schema integration; Semantic Relation ships; Word Net

1. Introduction

Big data refers to massive amounts of data collected over time that is difficult to analyze and handle using common database management tools. The analytical challenge is deriving meaningful information from data in petabyte and exabyte volumes. Big Data is typically described using the “3Vs”: volume, velocity, and variety. Volume refers to the amount of data made available from huge number of different resources. Velocity refers to the speed at which data is being collected and continuously made available. Variety refers to how data today is both varied and variable. Data sources are very heterogeneous both at the schema and at the instance levels. In other words, how data sources structure their data and at the instance level how the real world object is described.

In practical applications, data often does not come from a single source. Big data implementation requires

* Corresponding author. Tel.: +097144021464; fax: +97144021017.
E-mail address: basel.almourad@zu.ac.ae

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.258

handling data from various sources, in which data can be of heterogeneous of data types, representation, and semantic interpretation. This brings forth the challenge of data variety. The variety of data provides more information to solve problems or to provide better service. The question is how to capture the different types of data in a way that makes it possible to correlate their meanings. Identification of semantic relationships between data objects that are participating in a big data application is one of major activities and can be too complex or time consuming to be performed manually. On the lower level of the granularity it requires schema matching which is the problem of identifying elements of two given schemas that correspond or related to each other [1]. A number of traditional approaches have been proposed to infer semantic relationships between schema elements automatically. They can be classified into two main categories, *schema* and *knowledge based* approaches, reflecting the basis on which they detect semantic relationships. Both approaches have strengths and weaknesses, which will be discussed, in later section. In this paper we present our approach to the semantic correlation inferring. An algorithm that represents our approach is provided and explained. Section 2, presents the research related work. Section 3, discusses the role of meta data from the database design sematic point of view. Section 4, explains the role of WorldNet thesaurus in our research. Section 5, explain and demonstrates our semantic correlation inferring algorithm. Section 6, concludes and discusses the future direction of our research.

2. Related Work

Developers today approach identification of semantic relationships between data objects by painfully defining and writing programs or by using existing traditional data warehousing approaches. The main approaches to detect semantic relationships between data objects can be classified into two categories, schema and knowledge based approaches, reflecting the basis on which they detect semantic relationships.

2.1. Schema Based Approaches

A common technique in schema based approaches is to reason about the meaning and resemblance of heterogeneous objects in terms of their meta-data representation in order to identify those that could be semantically related [2,3]. Promising techniques that have been developed include the use of heuristics to determine the similarity of objects based on the occurrences of related attributes in the objects and the percentage of related attributes [4]. The main limitation of the schema based integration approaches is that they mainly rely on the literal equivalency of the attributes. The consideration of semantic is barely considered. This was due to the early days of schema integration problem and the absence of knowledge bases and their use within the context of semantic heterogeneity and database integration.

2.2. Knowledge Based Approaches

These approaches employ techniques utilizing semantic knowledge (based on real-world experience) in the integration process. They start with pre-existing concept structures that model real-world knowledge, such as thesauri or ontologies [5]. The schema elements of the databases are semantically enriched by appropriate concepts from these concept structures before comparing the schema elements. The semantic relationships between schema objects are then inferred by determining how the corresponding concepts are linked through the concept structures. Siegel [6] adopted a rule-based approach to resolve semantic conflicts so that common concepts would not be concealed by these differences. Sheth et al [7] introduce the concept of semantic proximity in order to formally specify various degrees of semantic similarity among related objects in different application domains. This is based on the real-world context in which these objects are used. In our approach we decided to identify object semantic similarity by using a combination of both approaches (schema and knowledge based approaches). We tried to employ positive notations from both techniques. Our algorithm relies on WordNet thesaurus in determining the semantically related DB objects.

3. Database design and the role of meta-data from the semantic point of view

The designers of DB have to deal with different 'worlds': The Real world, the Conceptual world and the Represented world. When building schemas, the DB designers invent names to label schema elements. Since DBs are usually designed to model the real world, schema element names are normally natural language nouns chosen to provide a bridge between them and their corresponding conceptualization. This schema element names are normally called met-data or data about data. Meta-data is considered to be valuable resource for managing information sources [8,9]. In a typical DB integration exercise, a DB schema integrator needs to locate DB schema objects that are relevant to the user information requirement. An appropriate sub-set of schema objects could be selected. In this exercise various types of met-data are required to facilitate the above tasks. In our approach we use met-data of DB schema objects as heuristics to infer DB schema object correlation and use these correlations to measure their relationships. Our technique reasons about the meaning and resemblance of DB objects in terms of their meta-data representation in order to identify those that could be semantically related. The technique uses kind of heuristics to determine the correlation of objects based on the occurrences of related attributes in the objects and the percentage of related attributes. The basis of heuristic depends on attribute equivalence to determine whether objects are semantically equivalent. Wordnet thesaurus is used [10] to help automate the identification of semantically similar properties.

4. The Role of WordNet thesaurus

WordNet is a machine readable, on-line lexical database of English words (nouns, verbs and adjectives) [10]. The words are grouped into *synsets*: sets of synonyms (lists of synonym word forms that are interchangeable in some context). The words in a synset are selected so that they represent a single lexical concept. One of the main assumptions underlying WordNet is that the different meanings or senses of a given word can be conceived unambiguously by considering the other words in the corresponding synsets to which they belong due to their semantic relationships. Each WordNet relation is represented in a separate file by an operator name. Some operators are reflexive (i.e. the reverse relation is implicit). So, for example, if *x* is a hypernym of *y*, *y* is necessarily a hyponym of *x*. Semantic relations are represented by a pair of *synset_ids*, in which the first *synset_id* is generally the source of the relation and the second is the target. If the pair *synset_id*, *w_num* is present, the operator represents a lexical relation between word forms. In our technique we consult synset predicates to detect synonyms that may be used when identifying similar objects. The synset predicate has the following syntax:

s (synset_id, w_num, 'word', ss_type, sense_number, tag_state)

Where an **s** operator is present for every word sense in WordNet and w_num specifies the word number for this word in the synset. For example, to find whether *scholar* and *student* are synonyms or not we use the following predicate: *s(X,_, scholar, _, _, _),s(X, _,student, _, _, _)*

The result is either true or false depending on whether scholar and student belong to the same synonym set or not.

5. Semantic correlation inferring algorithm

A semantic heterogeneity difference between relations of two different DBs is usually of interest only when the relations have some sort of resemblance so that they can be integrated in a way that satisfies their context and fits the user requirements [11].

Various types of semantic relationship are possible between different relations (e.g. equivalent, overlap, inclusion, disjoint). There are a number of classifications reported in the literature. For example, they can be classified according to the real world objects they represent [12], or they can be classified with respect to a *concept space* constructed for a federation [13]. We presume that two relations are *Semantically Related* when they have corresponding intended *Real World Semantics (RWS)* for some universe of discourse and *Semantically Incompatible* when they are not semantically related. The real-world semantics of a relation R, RWS(R), is defined as the set of objects in the real world defined by R's DB schema definition. As we cannot depend on the extension of relations in reality, we use relation properties (Property here refers to an attribute and its data type) as the basis for relation

comparison, assuming that the properties represent the intended meaning of the relations. In a real life application, complete semantic or syntactic equivalence between the related components of databases being integrated should not be expected to occur very often. Therefore, we adopt the notion of similarity rather than equivalence between database properties as the basis of our research. To detect whether two relations are similar, we designed a heuristic algorithm known as *Relation Similarity Detector (RSD)*. The RSD quantifies the measure of similarity between two relations in disparate DBs according to a hierarchical aggregation of similar properties. If the measure exceeds or equals a certain threshold (which can be altered by the user), then we consider the two relations to be similar. The RSD algorithm consists of two main functions: Relation Name Similarity Factor (RNSF) and Relation Property Similarity Factor (RPSF) (see the algorithm in Table 1). and one auxiliary (Equal).

The result of applying each function is a value in [0, 1]. W_{RNSF} and W_{RPSF} are RNSF and RPSF similarity weights respectively. A high value forces the RSD heuristic algorithm to detect only the relations that have highly similar properties and ultimately very close relations and this could ignore some potentially related relations. However, a low value could mean the heuristic algorithm has to reject many relations as non close which might be related.

RNSF procedure determines whether two relations have equal or similar names based on WordNet. For this purpose, it consults *EqualName* Function in the Equal procedure (described later). The result of applying RNSF is a value in [0, 1] (see the algorithm in Table 1).

Table 1. The algorithms of RSD and RNSF functions

RSD (Relation 1, Relation 2)
RNSF (Relation1_name, Relation2_name) = get the value of relation *name* similarity factor
RPSF (Relation 1, Relation 2) = get the value of relation *property* similarity factor
Return (RNSF * W_{RNSF} + RPSF * W_{RPSF})
	RNSF(Relation1_name, Relation2_name) //use EqualNam Algorithm
	If Relation1_name= Relation2_name Then
	Return 1
	If Relation1_name is synonym to Relation2_name Then
	Return 0.5
	If Relation1_name is suffix or prefix to Relation2_name Then
	Return 0.3
	If Relation1_name is subname of Relation2_name Then
	Return 0.3
	Otherwise Return 0

RPSF calculates the relation property similarity factor by dividing equivalent properties to the average number of properties in both relations. It starts by initiating two arrays Relation1_Property and Relation2_Property with properties extracted from Relation1 and Relation2, respectively. Each element of the array has the form [PropertyName, PropertyType] (e.g. [name,string]). RPSF then uses two loops to iterate both arrays and detect whether a property in Relation1_Property array is equal to a property in Relation2_Property array by using the Equal function (described later). If two properties are detected as equal, they are stored in Eq_Prpt, which is a list holding equivalent properties. If a property from Relation1 or Relation2 is not detected to have a similar property, it is stored in Eq_Prpt list with a property equivalent to null. The variable q is used to hold the number of pairs of equal properties in the list and it is used to calculate RPSF.

The findings of this algorithm are stored in the *property knowledge* base file which is a meta knowledge base. This knowledge base is used by different functions of the algorithm. The following is n example of these facts:

Pr_eq(db1, address, [street_name, string], primitive, db2, address, [street, string], primitive, 0.44)

The fact shows that the *street_name* attribute of relation *address* in database *db1* (which has a primitive string type) is equivalent to the primitive *street* attribute of *address* relation in database *db2* and it is given a 0.44 value. The Equal algorithm detected this similarity. The following fact shows that the *area* attribute of the *address* relation in *db1* has no equivalent attribute in relation *address* in *db2* and for this reason, it is considered similar to null and has been assigned a zero value

pr_eq(db1,address,[area,string],primitive,db2,address,null,_, 0)

Table 2. The Algorithm of RPSF function

```
RPSF(Relation1,Relation2)
q = 0; // Counter for number of equivalent properties
Eq_Prpt = {}; // List to hold equivalent properties
n = Number of Properties in Relation1; m = Number of Properties in Relation2;
Relation1_Property[n] = Get_RelationProperty(Relation1);
Relation2_Property[m] = Get_RelationProperty(Relation2);
While n <> 0 Do
    begin
      While m <> 0 Do
        begin
          If n = 0 Then exit \\ inner loop
          If Equal(Relation1_Property[n],Relation2_Property[m]) > threshold Then
          Begin
              add(Relation1_Property[n]= Relation2_Property[m], Eq_Prpt); q = q +1;
              remove(Relation1_Property[n], Relation1_PropertyList); n = n −1;
              remove(Relation2_Property[m], Relation2_PropertyList); m = m −1;
          end{if}
        end
    add(Relation1_Property[n]= null, Eq_Prpt);
    If m = 0 Then exit // outer loop
end{while}
// Continue
If n <> 0 Then
   Begin
    For i: = 1 to n Do
      Begin
          add(Relation1_Proeprty[i]= null,Eq_Prpt);
    end {for}
end{if}
If m <> 0 Then
    Begin
    For i =1 to m Do
      Begin
          add(Relation2_Property[i] = null,Eq_Prpt);
      End{for}
End{if}
Return (q/((n+m)/2))
```

5.1. The Equal Function

The Equal function (see the algorithm in Table 3) detects whether two properties are similar or not. This uses two functions *EqualName* and *EqualType*. The EqualName function checks whether two properties have similar names by checking different criteria (e.g. one property is a suffix, prefix or sub name to the second property). The WordNet database is consulted to detect whether two properties are synonyms or not. Different weights are given for the different finding. The result of applying EqualName is a value in [0, 1]. The EqualType function checks whether two properties have similar types or not. If both types are primitive, EqualType checks whether the two types are equivalent or are members of similar types. For example, varchar, varchar2, char, char(n) are compatible because they are members of a character data type set. If one of the types is a non-primitive type, then EqualType considers both types to be non-similar. An important case is when the two types are non-primitive types (user defined data type), where the attributes represent a relationship (generalization/association). In this case EqualType calls RSD (a recursive call) to detect whether the two data types are similar or not. The result of applying EqualType is a value in [0, 1]. W_{PNS} and W_{PTS} are the weights for EqualName and EqualType respectively.

Table 3. The Algorithm of Equal, Equal Name and Equal Type functions

```
Equal(Property1, Property2)
roperty1_name, Property2_name)
Property1_type, Property2_type)
    If PNS * W_PNS = 0 then
      Return 0
    else
      Return PNS * W_PNS + PTS * W_PTS
```

```
EqualName(Property1_Name, Property2_Name)
If Proeprty1_name = Property2_name
 Return 1
prefix to Property2_name
Return 0.3
 to Property2_name
Return 0.5
If Property1_name is synonym to Property2_name
Return 0.5
```

```
EqualType(Property1_type, Property2_type)
 If Property1_type = Proeprty2_type Return 1
Property1_type and Property2_type are member of similar types Return 0.5
Property1_type or Property2_type are no similar type Return 0
Property1_type and Property2_type are non-primitive type Return
RSD(Property1_type,Property2_type)
//Recursion
```

5.2. Example

Table 4 shows the similarity properties of Flight and Journey which are two relations from tow travel agent databases DB1 and DB2 respectively. The outcome of running RNSF procedure is 0.5 as the relations Flight &

Journey have different names, the Word Net thesaurus is consulted and it is found that Flight & Journey are synonyms. RPSF factor reflects the number of similar properties. Assuming that W_{PNS} (the weight of PNS) is 0.8, W_{PTS} (the Weight of PTS) is 0.2 and the PS threshold value is 0.4.

Table 4 Table 1similarity property of relations DB1_flight and DB2_journey

DB1_Flight		DB2_Journey		PS	>=0.4
Attribute	Data Type	Attribute	Data Type		
Departing_from	Varchar2(30)	Leaving_from	Varchar2(30	0.3*0.8+1*0.2 = 0.44	Yes
Going_to	Varchar2(30)	Arraiving_to	Varchar2(30)	0*0.8+ 1*0.2 = 0.2	No
Departure_date	Date	Departure	Date	0.5*0.8+1*0.2= 0.6	Yes
Returning_date	Date	Return	Date	0.5*0.8+1*0.2= 0.6	Yes
Rate	Number	Price	Number	0.5*0.8+1*0.2 = 0.6	Yes
Duration	Number	-	-	= 0	No
-	-	Class	Varcha2(15)	= 0	No

The average number of properties across both relations is 12/2= 6 therefore RPSF = 4/6 = 0.66. After calculating all factors, we can calculate the correlation of the Flight and Journey relations. If we assume that W_{RNSF} =0.4 and W_{RPSF} = 0.6, then the RSD algorithm outcome is: 0.5 *0.4 + 0.66*0.6 = 0.72. If we assume the RSD threshold value is 0.6 we can envisage that the relations Flight and Journey are correlated and can be integrated if we wish.

6. Conclusion & future work

Our approach to inferring database objects correlation uses resemblance function based on the schema meta-data such as names of schema element and data type. Word Net thesaurus is used to catch any potential semantic resemblance between schema objects. We are planning to extend the resemblance functions in future to capture semantic from schema instances. We are planning to include functions that support the identification of attribute with similar meaning by using aggregate instance information such as value distribution, term frequencies and average. We are also considering functions that rely on the relation instances.

7. References

1. Aminul I ,Diana I, [1] Iluju K. “Applications of corpus-based semantic similarity and word segmentation to database schema matching.” *The VLDB* 2008; 1293–1320
2. Palopoli L, Sacca D, Terracina G, Ursino D. “Uniform techniques for deriving similaritites of objects and subschemas in heterogeneous databases.” *IEEE Transactions on Knowledge and Data Engineering*; 2003; 271–294.
3. Rahm E, Bernstein PA. “A survey of approaches to automatic schema matching.” *The VLDB Journal* 10.4 2001; 334–350.
4. Sheth AP, Larson J, Cornelio A, Navathe,SB. “A tool for intergrating conceptual schemas and user views.” *4th International Conference on Data Engeneering*. Los Angeles, 1998; 176-182.
5. Ralyté J, Jeusfeld MA, Backlund P, Kühn H, Arni-Bloch N. “A knowledge-based approach to manage information systems interoperability.” *Information Systems* 7-8; 2008; 754-784.
6. Siegel M, Madnick S. A metadata approach to resolving semantic conflicts. In : International Conference on Very Large Databases, Barcelona, Spain, 1991; 133-145
7. Sheth A, Gale S, Navathe S. On Automatic reasoning for schema integration. International Journal of Intelligent and Cooperative Information Systems;1993; 2(1) 23-50.
8. Begg, T.: Database Systems: A Practical Approach to Design, Implementation and Management 5th edn. Addison Wesley; 2009
9. Kantere V, Tsoumakos D, Sellis T, Nick R: GrouPeer: Dynamic clustering of P2P databases. Information Systems; 2009; 34(1); 62-86.
10. Miller G. : Wordnet: Alexical database for English. Communication of the ACM; 1995.
11. Al-Mourad MB, Gray WA, Fiddian NJ. Semantically Rich Materialisation Rules for Integrating Heterogeneous Databases. In : Database: Enterprise, Skills and Innovation 3567. Springer-Verlag Berlin Heidelberg; 2005.
12. Blanco J, Illarramendi A, Goni A. Building a federated relational database system: An Approach using a knowledge-based system. International Journal on Intelligent and Cooperative Information systems; 1994; 3(4); 415-455.
13. Jarke M, Gallersdorfer R, Jeusfeld M, Staudt M.: Concept-Base - a deductive object base for meta data management. Journal of Intelligent Information Systems;1994; 3, 167-192.