Vol 5. No. 4, June 2012 ISSN 2006-1781
African Journal of Computing & ICT

www.ajocict.net

State-of-the Art: A Comparative Analysis of Ontology Matching Systems

[1]O. Iroju
Computer Science Department, Adeyemi College of Education, Ondo State, Nigeria.
irojuolaronke@gmail.com .

A. Soriyan, I. Gambo & Ikono, R.
Computer Science and Engineering
Obafemi Awolowo University
Ile-Ife, Osun State, Nigeria.
hasoriyan@yahoo.com, ipgambo@gmail.com, rhoda_u@yahoo.com

[1]*Corresponding Author*

ABSTRACT
Ontologies play an important role in resolving semantic heterogeneity in distributed and open systems such as the semantic web, information integration, peer-to-peer communication and agent communication. However, these systems adopt diverse ontologies which raise the semantic heterogeneity problem to a higher level. Consequently, the relationships, as well as the similarities between the terms in the different ontologies are usually matched to achieve interoperability by resolving the problem of semantic heterogeneity. Currently, there are different ontology matching systems which are used to determine the semantic correspondences among pairs of ontologies. This paper reviews and compares a number of ontology matching systems as well as examines their strengths and limitations.

Keywords: Ontology, ontology matching, ontology matching systems, heterogeneity, interoperability

1. INTRODUCTION

The proliferation of the Internet in recent times has facilitated the distribution and the accessibility of all kinds of information. This information is usually unstructured and presented heterogeneously. As a result of this, it becomes quite difficult for computers to capture the semantics of the information. Ontologies have been developed to describe the semantics of data, provide semantics for annotations in Web pages and ensure that the knowledge in computational applications is explicit. However, the popularity of the internet has resulted in a continuous growth in the numbers and sizes of available ontologies. These ontologies usually cover overlapping domains and this has resulted in an increased level of heterogeneity. This problem of heterogeneity can be resolved by matching the diverse ontologies [1]. Ontology matching finds correspondences between semantically related entities of the ontologies in order to reduce heterogeneity between them. These correspondences may stand for equivalence as well as other relations, such as subsumption, or disjointness, between ontology entities. Ontology matching results are referred to as alignments. Ontology alightment expresses with various degrees of precision the relations between the ontologies.

African Journal of Computing & ICT Reference Format:

O. Iroju, A. Soriyan, I. Gambo & R. Ikono(2012). State-of-the Art: A Comparative Analysis of Ontology Matching Systems. Afr J. of Comp & ICTs. Vol 5, No. 4. pp 81-93

Alignments can be used for tasks, such as ontology merging, data integration and translation, data warehousing and for query answering on the web. Ontology matching enables the knowledge and data expressed in the matched ontologies to interoperate. This paper therefore presents the concepts of ontology and also discusses mismatching between ontologies. The paper also addresses ontology matching, as well as presents and compares existing ontology matching systems with the aim of revealing their strengths and weaknesses. This will aid in determining which system produces a high quality result.

2. ONTOLOGY

Ontology is derived from the two Greek words namely ontos which means “to be” and logos meaning “word” [2]. The term ontology is also derived from the philosophy field which refers to a systematic approach to explain the existence of things in the world [3]. Ontology typically provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary [1]. They are generally used to provide a uniform conceptualization of terms [4]. The most popular definition of ontology is given by Gruber (1993). Gruber defines ontology as a formal explicit specification of a shared conceptualization [5]. In this definition, formal refers to the meaning of the specification which is encoded in a logic-based language, explicit means that concepts, properties, and axioms are explicitly defined, shared indicates that the specification is machine readable, and conceptualization models how people think about things of a particular subject area.

describes a specific domain of interest by including concepts and the relations among them. In general, ontology describes a certain domain by dividing it into several concepts or entities and describing the relations between those concepts. Concept also called a class is the major component of ontology. It is usually described by several attributes, which are concrete data fields such as data type properties. Ontologies may also contain instances, that is, concrete values of concepts. Ontology usually takes the form of a hierarchy of symbols. These symbols represent the concepts of a particular domain. The hierarchy is sometimes referred to as taxonomy and symbols are referred to as concepts, vocabulary or terms. An ontology includes a set of axioms to restrict the possible interpretations of its symbols. These axioms express the constraints that the symbols involved in those axioms must comply to. These axioms relate one symbol with the other symbols of the ontology. They restrict the possible interpretations for that symbol. Therefore, the most important part of ontology is the semantics associated with its symbols, usually referred to as the content of the ontology. The content of an ontology is constrained through its set of axioms [6]. A formal notation of ontology is as given below: An ontology is a tuple

$O = < C ; A; I; P > \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.....(1)$

such that:

$C = \{c_1, c_2\ldots\ldots\ldots\ldots c_k\}$ …….. (2)
is the set of concepts.

$A = \{A(c_1), A(c_2)\ldots\ldots\ldots A(c_k)\}\ldots\ldots\ldots$ (3)

with
$A(c_l) = \{a_{l1}\ldots\ldots\ldots a_{ln}\}\ldots\ldots\ldots\ldots\ldots\ldots\ldots.....(4)$

being a set of attributes assigned to a concept cl.

$R = \{r_1, r_2\ldots\ldots\ldots r_m\}\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$

R it a set of f relations
$I = \{i_1, i_2\ldots\ldots\ldots\ldots\ldots i_n\}\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(6)$

is a set of instances assigned to a concept C.

Ontologies are widely used in Knowledge Engineering, Artificial Intelligence and Computer Science; in applications related to areas such as Knowledge Management, Natural Language Processing, e-Commerce, Intelligent Information Integration, Bio-Informatics, Health Informatics, Education; and in new emerging fields like the Semantic Web. The goal of ontology in computer science is to add semantics to data in such a way that an application can classify the information [7]. Ontologies can be classified according to the issue of the conceptualization. These include general or upper level ontologies, domain ontologies and application ontology. General or upper level ontology classifies the different categories of entities existing in the world. General notions which are independent of a particular problem or domain are represented in these ontologies.

Knowledge defined in this kind of ontologies is applicable across domains and includes vocabulary related to things, events, time, space, mereology. Domain ontologies are more specific ontologies. Knowledge represented in this kind of ontologies is specific to a particular domain. These ontologies describe vocabulary related to a generic domain. They provide vocabularies about concepts in a domain and their relationships or about the theories governing the domain. In the various areas of computer science, there are different data and conceptual models that can be thought of as ontologies. These include relational database schemas, entity-relationship models, taxonomy, directories, thesauri, XML schemas and formal ontologies [1].

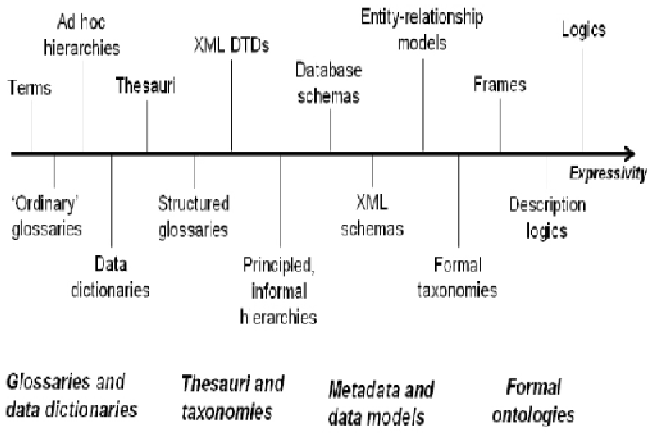

Fig 1. Various Forms of Ontology

Ontologies are usually represented in a structured way known as ontology languages, such that they can be easily processed by applications. Examples of ontology languages include Resource Description Format (RDF), Web Service Ontology Modeling (WSMO), Referent Modeling Language and Ontology Web Language (OWL). This structured way of representing the knowledge enables the extraction of the desired information for adequate applications. This also reduces the structural or syntactical heterogeneity of the knowledge [7].

3. ONTOLOGY HETEROGENEITY

There are various forms of ontological heterogeneity [1]. These include:

3.1 Syntactic heterogeneity

This occurs when two ontologies are not expressed in the same ontology language. A typical example is when ontologies are modeled using different knowledge representation formalisms such as OWL and RDF.

3.2 Terminological heterogeneity

This occurs when different terms represent the same entities in different ontologies. This can be caused by the use of different natural languages. An example of this type of heterogeneity is surname and family name or paper and article which are synonymous terms.

3.3 Conceptual heterogeneity or semantic heterogeneity

This is the collective term for different possibilities to model a domain. It can be divided into three sub problems. These include the difference in coverage, granularity and perspective. Coverage differences occur if ontologies are written from the same point of view, that is, they are written in the same context and with comparable vocabulary, but the part of the domain that is described differs and there are only overlapping parts. Difference in granularity occurs when the same section of the domain is described but the depth of details is not equal. If the point of view from which ontology is designed differs, there is a difference in perspective.

3.4 Semiotic heterogeneity or pragmatic heterogeneity

This is caused by the subjective interpretation of the used terms by humans. Terms can be considered regarding their context, such that terms with the same semantics are interpreted in different ways.

4. ONTOLOGY MATCHING

Ontology matching also referred to as ontology mapping is the process of setting up conjunction between different ontologies without changing the original ontology, so that both sides can obtain a common understanding of the same object [8]. It can also be defined as a process of finding a corresponding entity with the same or the closest intended meaning between two or more ontologies [9]. It is established after an analysis of the similarity according to certain metrics of the entities in the compared ontologies.

Formally, the matching process can be defined as [7]

Match: $(O, P) \rightarrow (c_1, c_2)$…….....(7)

$c_1 \in O, c_2 \in P$ sim $(c_1, c_2) > t$ ….....….(8)

where

O and P are the ontologies, c_1 and c_2 are the concepts or entities of the ontology, sim is the similarity function between the two entities and t is the similarity threshold.

Ontology matching can be represented as a function that matches two input ontologies O and O' by using a previous alignment A, a set of parameters and several other resources such as a knowledge or domain specific thesauri. As a result, an alignment A' which represents the correspondences between the two input ontologies is produced.

A'=f (O, O', A, p, r)……...…….(9)

Ontology mapping process can be represented schematically as shown below.

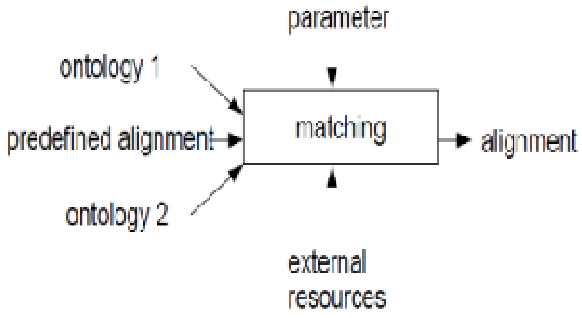

Fig 2. Ontology Mapping Process

The resulting alignment is defined as a set of correspondences, which represent relations between different entities. A correspondence can be described by a 5-tuple

<id, e,e',r, n>……(10)

Where
id is a unique identifier of the correspondence, e is an entity of ontology O, e' is an entity of O', r denotes an alignment relation such as equivalence (=), more general, overlapping and disjointness between the two entities and n gives a confidence value such as a similarity value. The notation above describes a 1:1 (one-to-one) correspondence, because exactly two entities are involved.

Ontology matching can also involve multiple ontologies due to the difference in granularity or coverage. This could result in 1: n correspondence (one-to-many) when an ontology is mapped with multiple ontologies or n: m (many-to-many) correspondences when multiple ontologies are matched against one another. The ontology multiple matching process can be viewed as a function which involves a set of ontologies to match $\{O_1, \ldots O_n\}$, an input multi-alignment A, a set of parameters p and a set of oracles and resources r, which returns a new multi-alignment A' between the ontologies.

A'= f $(O_1, \ldots O_n, A, p, r)$……(11)

Ontology matching process has been categorized into three stages namely pre-matching stage, matching stage and post-matching stage [10].

The pre-matching stage involves feature engineering. This involves the transformation of two ontologies into a common format suitable for similarity computation. Syntactic normalization is also involved in the feature engineering task which involves natural language processing techniques such as tokenization, lemmatization and elimination.

Another task in the pre-matching stage is the determination of the next search step. This involves finding a matching candidate in the ontologies to be matched. The most common approach is to compare all entities of the first ontology with all entities of the second ontology. After the completion of the pre-matching stage, is the matching stage where the actual similarity computation is carried out to determine the similarity values between matching candidates.

The post-matching stage requires the aggregation of different similarity values into a single value for one candidate pair. The final task of the post-matching stage is the interpretation of the similarity value in order to derive the best matching pair(s) among concepts in the first ontology and a set of concepts in the second ontology. These five tasks of ontology matching iterate until no new similarities are found [10].

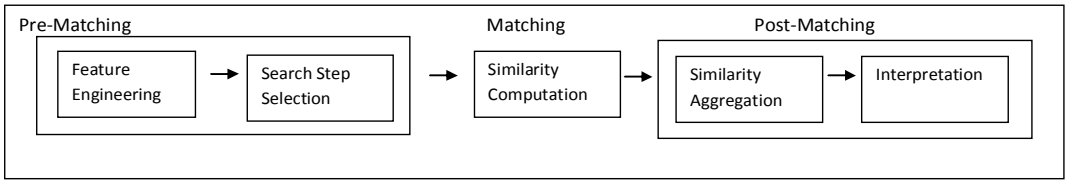

Fig 3. Ontology Mapping Tasks [10]

5. APPLICATION OF ONTOLOGY MATCHING

Ontology mapping has six central applications [11]. These include:

5.1 Ontology evolution

Ontology development, similar to software development, is developed in a distributed and collaborative manner. As a result of this, multiple versions of the same ontology often exist. A matching system is important in this process if there is a system that uses a version of the ontology and wishes to upgrade to another version. The major goal of ontology mapping in this process is to discover the correspondences between the ontology versions.

5.2 Information Integration

The most important application scenario for ontology matching systems is information integration [11]. This is because when multiple heterogeneous information sources are to be used for a common task such as query answering, the sources have to be matched and integrated. A matching system produces the connections between the sources by finding similar terms, concepts or attributes which provide the basis for the integration process. This is to ensure semantic interoperability among the systems to be integrated. In order to facilitate information integration, ontology mapping have been used in one of the three following ways.

1. *Single Ontology Approach:* All data source schemas are directly mapped to a shared global ontology that provides a uniform interface to the user. However, this approach requires that all data sources have nearly the same level of granularity, with the same perspective on a domain.
2. *Multiple Ontology Approach:* Each data source is represented in its own (local) ontology separately. Local ontologies are mapped to each other instead of mapped to a global one. In addition, additional representation formalism will be needed to define the inter-ontology mappings.
3. *Hybrid Ontology Approach:* This is a combination of the two previous approaches. It is t described as the most effective approach. In this approach, local ontology is built for each data source schema, without mapping to other local ontologies, but to a global ontology. New data sources can be easily added by not modifying existing mappings.

5.3 Peer-to-Peer Networks

In peer-to-peer networks, systems may exchange information. The peers can be totally autonomous; they might describe their data using different kinds of concepts or labels. In order to achieve meaningful information exchange among the systems, the different concepts or labels have to be matched.

5.4 Web Service Composition

Web services may present their interfaces using different languages. A matching process is needed to match the service descriptions as well as the inputs and outputs.

5.5 Agent Communication

Agents are computer entities characterized by autonomy and capacity of interaction. They communicate through speech-act inspired languages, such as the FIPA Agent Communication Language. The actual content of the message is represented in knowledge representation languages which often refer to some ontology. As a consequence, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages; however they would find it difficult to understand each other if they do not share the same content language and ontology. Thus, it is necessary to provide the possibility for these agents to match their ontologies.

5.6 Query Answering on the Web

The answering of queries is an important issue for the web. Search engines are used very often and they work with ontologies to refine queries. Each question that is posed by a user is translated into terms of the local ontology which is done by a matching process. Meta search engines translate a query into the terms of the different ontologies that are provided by the underlying search engines, collect the results and translate them back according to the original ontology.

6. ONTOLOGY MATCHING DIMENSIONS

There are several dimensions along which ontology matching algorithms are classified. These are primarily classified according to the input of the algorithms, the characteristics of the matching process, and the output of the algorithms [1].

6.1 Input Dimension

This dimension is concerned with the kind of input on which the algorithms operate. With the input dimensions, algorithms can be classified depending on the conceptual models in which ontologies are expressed such as XML, relational databases, entity relationship model and object oriented models or the kind of data that the algorithm exploits [1]. This could be schema level or instance level information or both.

6.2 Process Dimension

Process dimensions depend on the approximate or exact nature of the algorithm. Exact algorithms compute the absolute solution to a problem while approximate algorithms depend on performance rather than exactness. Process dimension could also be based on the interpretation of the input data. These can be classified into three basic categories which are syntactic, external, and semantic.

6.3 Output Dimensions

This dimension is concerned with the form of result or output produced by the algorithm. The alignment or result of the matching process could be a one-to-one (1: 1) alignment or an n-ary alignment (1:n or n:m). This dimension is also concerned with the kind of relations produced by the algorithm such as equivalence (=), subsumption, or incompatibility.

7. CLASSIFICATION OF MATCHING TECHNIQUES

There are various classifications of matching techniques. [1] classified ontology matching techniques into schema and structure based techniques. Other forms of ontology matching techniques include rule-based techniques, learning based approaches and instance based approaches.

7.1 Schema-Based Matching

Schema-based matching technique is based on three synthetic classifications. These include granularity or input interpretation which is also referred to as element- level or structure-level, basic technique layer and kind of input classification [1]. The granularity technique generally interprets the input information; the basic technique layer represents classes of elementary or basic matching techniques while the kind of input classification is based on the kind of input used by the matching techniques.

Schema matching technique can be at the element level or structure level. Mapping on the element-level implies that the elements (concepts, tables) are considered independently from their structure, that is, the relations to other entities are not taken into consideration. In contrast, structure-level compute correspondences by analyzing how entities appear together in a structure. At both levels, the syntactic techniques interpret the input data only considering the structure of the entities, while external methods use external knowledge like a thesaurus or human input. However, at the structure level, the semantic approaches exploit the meaning of the data [10].

8. CLASSES OF ELEMENTARY ONTOLOGY MATCHING TECHNIQUES AT THE ELEMENT-LEVEL

Ontology matching techniques at the element levels include string-based techniques, language-based techniques, linguistic resources, constraint-based techniques, alignment reuse, as well as upper level and domain specific formal ontologies.

8.1 String-based Techniques

These techniques are used to match names and name descriptions of ontology entities. They consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts. String matching techniques extensively used in ontology matching systems include prefix, suffix, edit distance, and n-gram:

1. *Prefix:* This test takes as input two strings and checks whether the first string starts with the second one. Prefix is efficient in matching cognate strings and similar acronyms. A typical example is int and integer

2. *Suffix:* This test takes as input two strings and checks whether the first string ends with the second one. A typical example is phone and telephone.

3. *Edit distance:* This distance takes as input two strings and computes the edit distance between the strings. That is, the number of insertions, deletions, and substitutions of characters required to transform one string into another.

4. *N-gram*: This test takes as input two strings and computes the number of common n-grams that is, sequences of n characters between them. For example, the trigram (3) for the string helium is hel, liu and ium.

8.2 Language-Based Techniques

These techniques consider names as words in some natural language such as English. They are based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words. Typical examples of language based techniques include

1. *Tokenization:* The names of entities are parsed into sequences of tokens by a tokenizer which recognizes punctuation, cases, blank characters, as well as digits

2. *Lemmatization:* The strings underlying tokens are morphologically analyzed in order to find all their possible basic forms.

3. *Elimination:* The tokens that are articles, prepositions, and conjunctions, are marked to be discarded.

8.3 Constraint-Based Techniques

These are element level techniques which deal with the internal constraints that are applied to the definitions of entities, such as types, cardinality of attributes, and keys [15]. Typical examples of this approach include:

1. *Data Type Comparison*: This involves comparing the various attributes of a class with regards to the values of their data types.
2. *Multiplicity comparison:* This involves the collection of attribute values by a particular construction set or list on which cardinality constraints are applied.

8.4 Linguistic resources

Linguistic resources involve the use of a common knowledge or domain specific thesauri to match words based on linguistic relations between them such as synonyms and hyponyms. This technique usually considers ontology entities as words of a natural language. Typical examples of domain specific thesauri include WorldNet, an electronic lexical database for English and other languages and Unified Medical Language Systems.

8.5 Alignment Reuse

This technique is an alternative way of exploiting external resources by recording alignments of previously matched ontologies. The alignment reuse is motivated by the fact that many ontologies to be matched are similar to already matched ontologies, especially if they are describing the same application domain. These techniques are particularly advantageous when dealing with large ontologies consisting of hundreds and thousands of entities. In these cases, large matched problems are decomposed into smaller sub-problems, thus generating a set of ontology fragments matching problems [1]. Then, reuse of previous match results can be more effectively applied at the level of ontology fragments rather than at the level of entire ontologies.

8.6 Upper level and domain specific formal ontologies

These techniques use upper level and domain specific formal ontologies as external sources of knowledge. Examples of the upper level ontologies are the Suggested Upper Merged Ontology (SUMO), and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). This technique has not been applied in ontology matching systems [1].

Ontology matching techniques at the structure level include graph based techniques, taxonomy based techniques, repository of structures and model based techniques.

8.7 Graph-Based Techniques

These are graph algorithms which consider the input as labeled graphs. The applications such as the database schemas or ontologies are viewed as graph-like structures containing terms and their inter-relationships. The process of finding the correspondences between elements of the graphs corresponds to solving a form of the graph homomorphism problem.

8.8 Taxonomy-based techniques

These are also graph based algorithms which consider only the specialization relation, that is, they only consider the "is-a" relations. Concepts linked with this relation are considered similar, their neighbours are as well considered similar. This consideration can be exploited in different ways:

1. *Bounded Path Matching:* Bounded path matchers take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and then identify similar terms.
2. *Super/Sub Concepts Rules:* These matchers are based on rules capturing the fact that if super-concepts are the same, the actual concepts are similar to each other. If sub-concepts are the same, the compared concepts are also similar [16].

8.9 Repository of Structures

The repository of structures stores all ontologies and the pair wise similarities between them. The goal of this structure is to find similar ontologies for each input ontology, such that the alignment can be reused.

8.10 Model-Based

These are algorithms which handle inputs based on its semantic interpretations. Examples are propositional satisfiability (SAT) and Description Logics (DL) reasoning techniques. Propositional satisfiability involves the decomposition of a graph matching problem into the set of node matching problems. Each node matching problem, that is, the pairs of nodes and the possible relations between them is translated into a propositional formula of form:

$$\text{Axioms} \rightarrow \text{Rel (context1, context2)} \quad \ldots\ldots\ldots(12)$$

The axiom encodes background knowledge which is used as premises to reason about relations holding between the nodes context1 and context2. A propositional formula is valid if its negation is unsatisfiable. The unsatisfiability is checked by using SAT solvers. SAT solvers are correct and complete decision procedures for propositional satisfiability which can be used for an exhaustive check of all the possible correspondences. This same procedure can be carried out within description logics. However, in Description Logics, the relations are expressed as a function of subsumption [15].

Other forms of ontology matching techniques include:

1. *Rule-Based Techniques*: Rule-based techniques work with schema-level information, such as entity names, data types and structures. Some examples of rules are that two entities match if their names are similar or if they have the same number of neighbor entities.
2. *Learning-based Techniques:* Learning-based approaches can be used at both the instance-level as well as the schema level. The learning based approaches perform ontology matching by comparing value formats and distributions of data instances underlying the entities under consideration.
3. *Instance-Based Techniques:* Instance based approaches are considered whenever the schemas to be matched use different terms to describe the same real-world-concept. Instance-based techniques can be used to improve the

effectiveness of the schema based approaches. Instance based approaches are classified into two. These include the linguistic and the constraint-based methods. Linguistic-based approaches use information retrieval techniques to gain information about word frequencies and combination of words. The use of value ranges, data types or the appearances of character patterns characterize the constraint-based approaches.

9. OVERVIEW OF ONTOLOGY MATCHING SYSTEMS

The matching of heterogeneous information sources has been widely addressed in the last decades. Some of these systems include:

9.1 Automated Semantic Matching of Ontologies with Verification (ASMOV)

ASMOV combines a comprehensive set of element-level and structure-level measures of similarity with a technique that uses formal semantics to verify whether computed correspondences comply with desired characteristics. The ASMOV process is an iterative process divided into two main components: similarity calculation, and semantic verification. ASMOV receives as input two ontologies in OWL-DL format to be matched, and an optional input alignment, containing a set of predetermined correspondences. The first stage is the similarity calculation process which computes a similarity value between all possible pairs of entities, one from each of the two ontologies, and uses the optional input alignment to adjust any calculated measures.

ASMOV obtains a calculated similarity measure as a weighted average of four similarities which include a lexical or terminological similarity, using either an external thesaurus specially WordNet and UMLS or string comparison; two structural similarities, a relational or hierarchical similarity, which uses the specialization relationships in the ontology; an internal or restriction similarity, which uses the established restrictions between classes and properties, and an extensional similarity, which uses the data instances in the ontology. The similarity among the entity pairs belonging to two ontologies is stored in a similarity matrix. From this similarity matrix, a pre-alignment is extracted, by selecting the maximum similarity value for each entity using greedy algorithm.

This pre-alignment is passed through a process of semantic verification, which eliminates correspondences that cannot be verified by the assertions in the ontologies, resetting the similarity measures for these unverified correspondences to zero. This process results in a semantically verified similarity matrix and alignment, which are then used to evaluate a finalization condition. If this condition is true, then the process terminates, and the resulting alignment is final. The drawback of this system is that when the given two ontologies are dissimilar effectiveness is decreased. Also, semantic verification system eliminates too many alignments leading to less recall. ASMOV also needs polynomial time for each iteration which leads to inefficiency.

9.2 Cupid

Cupid is a generic schema matching algorithm that combines element and structural matching techniques. It also includes automated linguistic-based matching. It takes as input two schemas which are encoded as graphs. The nodes represent schema entities and are traversed in a combined bottom up and top down manner. Cupid consists of three stages. These are the linguistic matching phase, the structural matching phase and the matching generation phase. The first stage is the linguistic matching which matches individual schema elements based on their names, data types and domains. It employs the use of a thesaurus to help match names by identifying short-forms. The linguistic matching consists of three steps which include normalization, categorization and comparison. The normalization process involves three stages which includes tokenization, elimination and tagging. Tokenization is the process where the names of the entities are parsed into tokens by a customizable tokenizer. In the elimination phase, tokens that are articles, prepositions or conjunctions are marked to be ignored during comparison. In the tagging phase, a schema element that has a token related to a known concept is tagged with the concept name. Next, Cupid clusters schema elements belonging to the two schemas into categories. The purpose of categorization is to reduce the number of element-to-element comparisons. In the comparison phase, the similarity of two name tokens is looked up in a synonym and hypernym thesaurus. Each thesaurus entry is annotated with a coefficient in the range of [0,1] that indicates the strength of the relationship. The linguistic similarity of each pair of elements from compatible categories is computed. Linguistic similarity is based on the name similarity of elements, which is computed as a weighted mean of the per-token-type name similarity. The linguistic similarity is computed by scaling the name similarity of the model elements by the maximum similarity of categories to which they belong.

The result of this phase is a table of linguistic similarity coefficients between elements in the two schemas. The similarity is assumed to be zero for schema elements that do not belong to any compatible categories. The second phase is the structural matching of schema elements based on the similarity of their contexts. The structural matching depends on the linguistic matches calculated in phase one. The result is a structural similarity coefficient for each pair of elements. A weighted similarity which is the mean of linguistic and structural similarity is calculated. The third phase is the mapping generation phase. Mapping elements are generated using the computed linguistic and structural similarities. This is done by choosing the pairs of schema elements with maximal weighted similarity. One of the drawbacks of Cupid is the limited means of capturing user interaction.

9.3 OWL Lite Alignment (OLA)

OWL-Lite Alignment (OLA) is an ontology alignment tool that is jointly developed by the teams at DIRO, University of Montréal and INRIA Rhône Alpes. OLA is dedicated to the alignment of ontologies expressed in OWL, with an emphasis on its restricted dialect of OWL, called OWL-Lite. OLA is based on both schema and instance mapping techniques. OLA takes as inputs two ontologies.

These ontologies are parsed and their entities and their relationships extracted. The entities and their relationships are represented in a graph structure referred to as OL-Graph. An OL-Graph is a labeled graph where vertices correspond to OWL entities and edges represents the inter-entity relationships. The set of different vertex categories represents the class, relation, property, property instance, and the data type, data value, and property restriction label. The relationships expressed in the OL-Graph are specialization between classes or relations, instantiation between objects and classes, property instances and properties, values and data types, attribution between classes and properties, objects and property instances, restriction expressing the restriction on a property in a class and valuation of a property in an object. These graph structures produce the constraints for expressing a similarity between the elements of the ontologies.

The similarity between nodes of the graphs depends on the category of node considered, such as class. It also takes into account all the features of this category, such as super classes. The lexical similarity between the entity pairs is calculated. OLA relies on WordNet 2.0 for comparing identifiers by applying a measure of relatedness between two terms. The lexical similarity mechanisms retrieve the sets of synonyms for each term and a normalized Hamming distance is then applied to these sets. A variant of the substring distance is used to establish a default similarity value for identified pairs. These distances are linearly aggregated. This process is iterated until no improvement is produced. Thus, an alignment is generated which satisfies some additional criterion on the alignment obtained and the distance between matched entities.

9.4 Risk Minimization based Ontology Mapping (RiMOM)

RiMOM takes two ontologies as input. The mapping process consists of six phases, which include

1. *Preprocessing*
   RiMOM generates the description for each entity in the ontologies. It then calculates the two similarity factors which are the label and structure similarity factors.

2. *Linguistic-Based Ontology Alignment*
   In this step, multiple linguistic-based strategies are executed. The strategies used at this stage are the edit distance and vector based strategy. Each of these strategies uses different ontological information and obtains a similarity result for each entity pair.

3. *Similarity combination*
   This step combines the similarity results obtained by the selected strategies.

4. *Similarity propagation*
   This step considers structural similarity. Three similarity propagation strategies, namely, Concept-to-Concept, Propertyto- Property, and Concept-to-Property were used.

5. *Alignment Generation and Refinement*
   This step fine tunes and outputs the alignment result.

The drawback of RiMOM is its inefficiency for dealing with large scale ontologies. RIMOM consumes long time and large amount of memory since it does not incorporate search space reduction techniques like early pruning, partition of ontology or parallel technique.

9.5 GLUE

Glue is an ontology mapping system that exploits both schema and instance-level information. Glue takes as input two ontologies and finds the most similar concept in the target ontology for each concept in the source ontology. Glue uses machine learning techniques to find the mappings. It first applies statistical analysis to the available data; it then generates a similarity matrix based on the probability distributions for the data considered. According to these probabilistic measurements, two base learning techniques are applied to build a similarity matrix expressing the prediction of semantic affinity between concepts. These learning techniques are the Content Learner and the Name Learner. The content learner uses a text classification method, called Naïve Bayes learning. The name learner is similar to the content learner but uses the full name of the instance instead of its content. Finally, Glue provides a Relaxation Labeler to search for the mappings that best satisfy the given domain constraints and heuristic knowledge. Glue may not be effective on short text content because it uses classifier to compute the similarity between entity names. In addition, Glue does not provide an interface for user interaction.

9.6 COMA/COMA++ (COmbination of Matching Algorithms) University of Leipzig

COMA/ COMA++ is a schema matching tool based on parallel composition of matchers. It is composed of a library of matching algorithms, a framework for combining the results obtained, and a platform for the evaluation of the effectiveness of the different matchers. COMA is made up of elementary matchers, five hybrid matchers which are combinations of elementary methods, and one reuse-oriented matcher. The elementary matchers implement string-based techniques such as affix, n-gram, edit distance and also employ the use of a thesauri look-up. The reuse-matcher reuse previously obtained results for entire new schemas or for its fragments. COMA presumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of match. COMA++ is built on top of COMA by elaborating in more detail the alignment reuse operation. The drawbacks of this method are as follows. COMA++ is developed for XML schemas and hence it is not suitable for complex ontology graph. It use relatively simple heuristic rules to partition the input schemas resulting often in too few or too many partitions and to find similar fragment pair only the root node features are used which will lead to less matching quality.

9.7 S-Match (Semantic Matching)

S-Match is an automatic matching system that takes two graph-like taxonomies as input and computes the strongest semantic relations holding between any pair of nodes from two taxonomies [12]. The match approach of S-Match is based on two notions. These are the notion of concept of a label and the notion of concept of a node.

The notion of concept is defined as the set of documents that reflects the meaning of the labels, while the concept of a node is defined as the set of documents classified under this node. Before executing the semantic matching process, the system uses natural language processing techniques to tokenize and lemmatize labels of the taxonomies. Subsequently, the labels are required to translate from natural language into a more precise internal format using WordNet. For complex concepts, logical connectives are used to replace tokens that contain prepositions, punctuation marks and conjunctions. To derive the strongest relationships, S-Match computes the concept of label matrix which contains relations between any two concepts of labels in the two taxonomies using both the string-based and linguistic resources matchers. After that, the information obtained from the matrix together with concepts of labels and concepts of nodes are codified into a set of complex propositional formulas. S-Match generates another matrix containing the strongest relations holding between concepts of nodes by applying model-based technique to process the propositional formulas. The major disadvantage of this system is that the input ontologies are not specific.

9.8 IF- Map

IF-Map is an automatic mapping method that uses model-based matching technique in its mapping process [12]. The technique is grounded on Channel Theory which provides a mathematical model to depict the flow of information in the connection channel between communities by means of tokens and types [13]. IF-Map associates ontology with a set of concepts, instances and relations and formalizes the mappings in terms of logic informorphisms IF-Map exploits a set of heuristics as well as string-based and taxonomy-based techniques to derive partial mappings between the concepts of the ontologies.

9.9 MAMA (Mapping And Merging Approach)

MAMA is based on the combination of various indexes which works at syntactic or lexical level, semantic level and structural level. This is to find the equivalence among the various components of the ontologies. The aim of the syntactic indexes is to detect the syntactical similarities among the various components of the ontology. Mama employs edit distance, trigram function, acronym, fingerprint, abbreviation, label and attributes as syntactic indexes. MAMA uses edit distance to calculate the likelihood between two words that label concepts in the ontology. The trigram function measures the number of similar trigrams that are in the words that label the concepts in the ontologies. The acronym index verifies if a word in the two comparing nodes is the acronym of the other. If it is true this index is 1, otherwise it is 0. The fingerprint index verifies if the word that describes a comparing node is in the other word that describes the other nodes. If the word is contained in the other one this index is 1, otherwise it is 0. The abbreviation index measures if a word that describes a node is an abbreviation of the other that describes the other comparing node. If the word is an abbreviation of the other one this index is 1, otherwise is 0. The label index measures if the two labels of the comparing nodes are equal. Also, in this case the index assumes value 1 if the nodes have the same label, and 0 if otherwise.

The Semantic indexes compare the ontologies from a semantic point of view. these indexes use structured knowledge as in WordNet. The set of semantic indexes used at this level include semantic similarity, granularity, synonym index, and derived index. The semantic similarity index compares two words according to the taxonomy defined in WorldNet from a semantic point of view. Granularity index measure the mutual position of the words representing the concepts of the ontology in the WordNet. Derived Index finds an adjective representing a node of an ontology, derived from the label of a concept that is in the other ontology in WordNet. This index assumes a value 0 or 1. The structural indexes compare the ontologies from a structural point of view. The indexes of this set are attributes, similarity index for properties, and similarity for entities. The attribute index measures the numbers of similar attributes between the two nodes. The property similarity index verifies the equality between the nodes by evaluating their properties. The similarity index for entities measures the numbers of similar entities. The three sets of indexes are combined in order to map the various nodes that are in the ontologies. The second step is the mapping among the relations while the last step is the mapping among the attributes. After this phase the mapping process among the ontologies is obtained. This approach only considers instance matching.

9.10 PROMPT

PROMPT is an algorithm developed by Noy and Musen in year 2000 for ontology merging and alignment (Interop, 2008). Prompt handles ontologies specified in OKBC compatible format. It starts with the identification of matching class names. Based on this initial step, an iterative approach is carried out for performing automatic updates, finding resulting conflicts, and making suggestions to remove these conflicts. PROMPT is implemented as an extension to the Protégé 2000 knowledge acquisition tool and offers a collection of operations for merging two classes and related slots. Prompt does not provide instance or structure matching and it does not deal with normalization process.

9.11 Anchor Prompt

Anchor Prompt is designed to augment existing methods, like PROMPT by determining additional possible points of similarity between ontologies. Anchor-PROMPT takes as input a set of pairs of related terms also called anchors from the source ontologies. From this set of identified anchors, Anchor-PROMPT produces a set of new pairs of semantically close terms. Anchor-PROMPT does this by traversing the paths between the anchors in the corresponding ontologies. A path follows the links between classes defined by the hierarchical relations or by slots and their domains and ranges. Anchor-PROMPT then compares the terms along these paths to find similar terms. The central observation behind Anchor-PROMPT is that if two pairs of terms from the source ontologies are similar and there are paths connecting the terms, then the elements in those paths are often similar as well. Therefore, from a small set of previously identified related terms, Anchor-PROMPT is able to suggest a large number of terms that are likely to be semantically similar as well. However, this approach does not work well when the source ontologies are constructed differently.

9.12 QOM (Quick Ontology Matching)

QOM is the iterative ontology matching algorithm which is a variation of the Naive Ontology Matching (NOM) algorithm dedicated to improve the efficiency of the system. The basic difference is QOM reduces the search space by eliminating less promising matching entity pairs using heuristic and dynamic programming approach with marginally compromising on effectiveness. QOM matcher consisting of the following steps to match the ontologies. The input to QOM is in RDF format. The process of determining whether two entities are same or not is based on the entity features. This step constructs the search space of ontology entities by computing the Cartesian product of entity pairs using entity features of both ontologies.

The entity pairs obtained will be pruned using heuristics strategies to reduce the search space. The selected matching pairs will be processed for similarity computation and their interpretation. The heuristics strategies are based on hierarchy of entities, entity neighbour to the alignments obtained from the previous iteration, labels of entities, random pick or combination of the above strategies. The similarity between the above selected entity pairs can be measured by wide range of similarity functions like Dice coefficient, Levenshtein's edit distance and cosine similarity value. Based on the feature of the entity any of the above similarity function can be chosen. To aggregate the similarity value QOM uses weighted average of similarity values for a given entity pair.

The weight of the similarity values is calculated by sigmoid function which emphasizes high individual similarities and deemphasizes low individual similarities. For the next iteration the matching pairs with similarity value less than a threshold and which violates bijectivity (one-to-one and onto constraint) of the matching are discarded. Initial iteration uses lexical knowledge while later iterations use structure knowledge for matching. The number of iteration required irrespective of ontology size based on their experiment is ten. Based on the evaluation QOM is nearly twenty times faster than NOM. The drawback of QOM is that, effectiveness is marginally lower than NOM since not all Cartesian entity pair is evaluated.

9.13 AnchorFlood Matching Technique

AnchorFlood is an ontology matching algorithm with a dynamic partition based matching which avoids the priori partitioning of the ontologies. The internal structure, external structure and Jaro-Wingler string distance is used in measuring semantic similarity value between the entity pairs. The initial input to the algorithm is the set of anchors, where an anchor is defined as an exact string match of concepts, properties or instances pair. Preprocessing of ontologies is carried out to normalize the textual contents of entities. The algorithm starts with an anchor and collects neighboring entity of the chosen anchor across ontologies. The segment pair constructed from the above collected neighbors is processed to produce alignment pairs. The process repeats until "either all the collected entity is explored, or no new aligned pair is found".

The anchor will be discarded as misalignment if the constructed segment of the anchor does not produce sufficient alignments. The drawbacks of this method are as follows. It ignores certain distantly placed aligned pairs since segments are constructed from neighbor of anchors. Semantic similarity between entities is not explored. Only exact string match of concepts, properties and instances are considered as anchors which lead to inefficient alignments.

9.14 Omen (Ontology Mapping ENhancer)

Omen uses a Bayesian Net and enhances existing ontology mappings by deriving missed matches and invalidating existing false matches. The Bayesian Net uses a set of meta-rules based on the semantics of the ontology relations that expresses how each mapping affects other related mappings. Nodes in the BN-graph represent individual pairs of matches.

9.15 Cluster-based Similarity Aggregation (CSA) for Ontology Matching

CSA is an automatic similarity aggregating system for ontology matching. The system has two main parts. The first part is the calculation and combination of different similarity measures. The similarities employed are string edit distance, WordNet based, profile, and structure and Instance based. The string edit distance measures the lexical feature of the entity's name. WordNet exploits the similarity between words occurrence in the entity's name. Wu and Palmer's algorithm were used for calculating the WordNet similarity. The Profile similarity makes use of the id, label, comments information contained in entity. The profile for a class takes their properties, instances into account. The profile for a property includes their domains and their ranges. The Structure similarity is calculated for class only.

This similarity measures the difference in the local structure of an entity. The instance based measure is similar to the Profile except that it utilizes the content of instances that belong to classes. Five similarity matrixes were created based on the similarity measures used. The weight for each similarity is estimated through a weight estimation process. Each similarity matrix of each basic measure was analyzed to find which one is actually effective and which one is not for alignment process. The weight estimation process is based on two information. First, for each single method, the process of finding a threshold for distinguish matching pairs from non matching pairs can be viewed as a binary classification problem. In this process the positive class contains matching pairs and the negative class contains non matching ones. Second, in one-to-one ontology alignment, the maximum number of matching pairs is equal to the minimum number of entities in the two ontologies.

Then if a single method is effective, its correspondent similarity matrix must have the two criteria: The matrix must have the ability of distinguishing matching pairs from non matching pairs and the number of matching pairs must approximate the minimum number of entities in the two ontologies. Base on these criteria, the weight estimation process was modeled using K-means algorithm to cluster the similarity values into two different classes (k = 2). The feature is the similarity value of each pairs of classes.

The cluster with higher mean represents the matching set, and the lower one represents the non matching set. Then values belonging to the non-matching was filtered leaving the similarity matrix with the higher values. Then the number of row that has value in the matrix was calculated. The rows represent the possible matching pairs. Finally, the weight is estimated by the ratio of the number of row over the number of matched values in the filtered matrix. The weight estimation for property similarity matrix is calculated in the same manner. The similarity combination can be defined as the weight average of the five basic measures. Similarity propagation considers the impact of structural information on the similarity between entity pair in the aggregated matrix. The intuition is that the more similar in structure two entities are, the more similar they are.

The final similarity matrix can be viewed as a bipartite graph with the first set of vertices as entities from source ontology and the second set of vertices are entities from target ontology. Thus, the alignment extraction can be modeled as the process of finding the mapping from the bipartite graph using a stable marriage problem algorithm. The two set of entities were modeled as sets of men and women. For each man and each woman, in the correspondence set, a list of priority of men and women is created based on their similarity value. The stable marriage algorithm is then applied to find the stable mapping between two sets. The result is the pre-alignment. Pruning is the final step of the system. In this step a proportion of entities pair that have low confidence was filtered out to increase the precision of the system. The result is the final alignment of the two ontologies. The major drawback of this system is that it does not consider instance matching.

9.16 Automatch

Automatch is an instance-based matching system which was developed at the University of Fairfax. It is originally designed for matching databases schemas but has been adapted to ontology matching tasks. Automatch uses knowledge bases which include example values to describe the attributes. For each attribute of a schema there is an attribute dictionary, which consists of possible instance values and probability estimates (scores). The probability estimates are calculated by applying a Bayesian learning algorithm. The values of the dictionary are chosen by following one of three statistical feature selection strategies: mutual information, information gain and likelihood ratio. The attributes directories are used to match the two input ontologies using the following algorithm: all attributes of the schemas are matched against the attribute dictionary and an individual match score is calculated.

The individual scores are summed up to obtain attribute pair scores, which provide a basis for the application of a minimum cost maximum flow algorithm that determines an optimal mapping. Automatch is time-consuming and it does not exploit schema information. Hence, entity pairs having the same name are not recognized without performing the complex instance-matching process. Additionally, Automatch cannot work with entities or ontologies that do not have instances].

9.17 Katrin, 2010

[7] Presented two novel instance-based matchers. A matcher that makes use of regular expressions or a catchword list to describe the instances of a concept was introduced. The catchword list was created manually by a domain expert. This provides a basis for creating a vectorial representation for each concept, the RECW vector. These vectors are compared by applying a cosine similarity measure and an alignment is produced. The second approach characterized the concepts by regarding their instance sets. A set of predefined features is determined. The features provide the basis for a pairwise comparison between attributes of different ontologies

A COMPARISON OF ONTOLOGY MATCHING SYSTEMS

Table I. COMPARISON OF ONTOLOGY MATCHING SYSTEMS

Ontology Matching Systems	Strategy Employed	Ontology Structure Considered	Type Of Correspondence	Ontology Language Supported	Limitations
ASMOV	Employed lexical or terminological similarity, as well as an external thesaurus specially WordNet and UMLS or string comparison	Concept and data instances	1:1	-	The drawback of this system is that when the given two ontologies are dissimilar effectiveness is decreased. Semantic verification system eliminates too many alignments leading to less recall. ASMOV needs polynomial time for each iteration which leads to inefficiency.
CUPID	Automated linguistic-based matching	Concept and relations	1:1	-	It has a limited means of capturing user interaction.
OWL Lite Alignment	schema and instance mapping techniques.	Concepts and relations	1:1	OWL-Lite	
RiMOM	linguistic-based matching	Concepts and relations	1:n	-	RiMOM is inefficient when dealing with large scale ontologies. RIMOM consumes long time and large amount of memory
GLUE	probability distributions	Concepts, instances, properties	1:1	-	Glue may not be effective on short text content because it uses classifier to compute the similarity between entity names.
COMA/COMA++	string-based techniques and thesauri look-up	Concepts	1:1	XML schemas	not suitable for complex ontology graph
S-Match	natural language processing WordNet	Concepts	1:1	-	the input ontologies are not specific
IF- Map	model-based matching technique string-based and taxonomy-based techniques	Concepts	1:1	RDF,Ontologua, Prolog	
MAMA	edit distance, trigram function, WordNet	Concepts and relations	1:1	-	does not consider instance matching.
PROMPT	linguistic-based matching	Concepts, instances, properties	1:1	OKBC	does not provide instance or structure matching and it does not deal with normalization process.
Anchor Prompt	linguistic-based matching	Concepts, instances, properties	1:1	-	does not work well when the source ontologies are constructed differently.
QOM	heuristics	Concept	1:1	RDF format	Reduced Effectiveness
AnchorFlood	string-based techniques and thesauri look-up	Concepts, instances and properties	1:1	-	Reduced Effectiveness
Omen	Bayesian Net	Concepts and relations	1:1	-	
CSA	string-based techniques WordNet	Concepts, instances and relations	1:1	-	
Automatch	Bayesian learning algorithm.	Concepts, instances	1:1	-	

10. CONCLUSION

Ontology matching is the primary means of resolving semantic heterogeneity. However, the diversity of ontologies raises semantic heterogeneity to a higher level. Ontology matching has been used to establish semantic correspondence in interoperation systems. This paper therefore appraised ontology, ontology heterogeneity, ontology matching, as well as ontology matching system. This was with a view of revealing the strengths and limitations of the ontology matching systems, therefore a future work will geared towards the development of a novel model for ontology matching.

REFERENCES

1. P. Shvaiko, "Iterative Schema-Based Semantic Matching," PhD Dissertation. International Doctorate School in Information and Communication Technology, University Of Trento, 2006.
2. P. Kenjige, "Usage of Medical Ontology in EA", PK Technologies, 2010.
3. Y. J., Wu, J. S. Lang, and F. H. Shang, A Similarity-Based Approach for Ontology Mapping," International Conference on Computer Science and Education. pp. 165-169, 2009.
4. A. Costa, S. S. Renata, G. Guizzardi,., and C. Jos´e, "CORe S: Context-Aware, Ontology-Based Recommender System for Service Recommendation," Department of Computer Science, UFES, Vitoria-ES, Brazil, ITC-irst, Trento-Povo, Italy and Laboratory of Applied Ontologies (ISTC-CNR), Trento, Italy, 2004.
5. T. Gruber, "Toward Principles for the Design of Ontologies Used For Knowledge Sharing," Stanford University, 1993.
6. B. Howard, and S.Helena, "Overview of Approach, Methodologies, Standards, and Tools for Ontologies," University of Florida, 2002.
7. S. Z. Katrin, "Instance-Based Ontology Matching and the Evaluation of Matching Systems," Inaugural-Dissertation, Department of Computer Science, Heinrich Heine University of Dusseldorf, Germany, 2010.
8. C. Rung-Ching, L. Bo-Ying, and B. Cho-Tscan, "Using Domain Ontology Mapping for Drugs Recommendation", Department Of Information Management, Chaoyang University Of Technology, Taiwan, 2009.
9. Interop ," State of the Art Report. Ontology Interoperability", Information Society Technologies, 2008.
10. M. Ehrig and S. Staab , "QOM - Quick Ontology Mapping", International Semantic Web Conference, vol. 3298, pp. 683–697, 2004.
11. J.Euzenat, and P. Shvaiko, "Ontology Matching", Springer-Verlag, Heidelberg, 2007.
12. Y. R Jean-Mary, P. Shironoshita, and M. Kabuka. "Ontology Matching with Semantic Verification", Journal Of Web Semantic, vol 7, No. 3, Pp. 235–251, 2010
13. F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "Semantic Schema Matching", Proceedings of the 13rd International Conference on Cooperative Information Systems (CoopIS), pp 347-365.
14. Y. Kalfoglou, M. Schorlemmer, M. Uschold, , A. Sheth, and S. Staab, "Semantic Interoperability and Integration", Dagstuhl Seminar, University of Southampton, UK, 2004.
15. J. Barwise, and J. Seligman, "Information Flow: The Logic of Distributed Systems", Cambridge Tracts in Theoretical Computer Science, Cambridge University Press. ISBN: 0-521-58386-1, 1997.
16. D. Rose and H. Stefan, "Comparison of Personal Ontologies Represented Through Conceptual Graphs", Proceedings Of The European Conference On Artificial Intelligence, pp 341–345, 1998.

Author's Briefs

Iroju Olaronke is a lecturer of the Department of Computer Science, Adeyemi College of Education, Ondo, Nigeria. Her research interest is on interoperability and ontology matching. She can be reached by phone on +2348034652253 and through E-mail irojuolaronke@gmail.com

Soriyan H. A. is an Associate Prof of Computer Science at the Obafemi Awolowo University Ile-Ife, Nigeria. Her research interest is in Information System, Health Informatics, and Software Engineering. She can be reached through hasoriyan@yahoo.com

Ishaya Gambo is a lecturer of Computer Science at the Obafemi Awolowo University Ile-Ife, Nigeria. He is pursuing a PhD degree in computer science. He has got a good number of publications in reputable journals and learned conferences. His research interest is in Information systems design and methodology, software engineering with emphasis on Software Architecture and software quality issues. E-mail ipgambo@gmail.com

Ikono R. N. is a lecturer of Computer Science at the Obafemi Awolowo University Ile-Ife, Nigeria. Currently on a PhD research programme in Computer Science. Her research interest is in Information System, Health Informatics, and Software Product Usability. She can be reached through rhoda_u@yahoo.com