Journal of Theoretical and Applied Information Technology
31st March 2019. Vol.97. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

BOOTSRAPPING INSTANCE-BASED ONTOLOGY MATCHING VIA UNSUPERVISED GENERATION OF TRAINING SAMPLES

[1]MANSIR ABUBAKAR, [2]HAZLINA HAMDAN, [3]NORWATI MUSTAPHA, [4]TEH NORANIS MOHD ARIS

[1, 2, 3,4]Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang Selangor Darul Ehsan
E-mail: [1]abubakar.mansir@auk.edu.ng, [2, 3, 4]{hazlina, norwati, noranis}@upm.edu.my

ABSTRACT

Training set is the key role player that can improve the performance of any classification task. Different techniques and methods are being applied to generate training set depending on its area of application. Researchers in data science and semantic web community use different kind of training sets generated to improve the performances of classifications and information retrieval capability. Operational Training Set Generator (TSG) should always solve a minimum of two issues; (1) it must address the computational cost in producing a reasonable outcome, thereby reducing the computational cost in the whole system. The run-time of TSG is near linear as in blocking approach and (2) it must produce the qualitative training sets. We use LogTfIdf as the cosine similarity function of two given vectors to produce Bag of Words (BoW); the tokenizer is developed to specially take care of delimiters that often come across URIs and other RDF essentials. We evaluated our UTSG on nine cross-domain benchmark ontologies publically available in OAEI website. The results obtained shows that our UTSG outperforms the two baseline TSGs previously developed to address similar problem.

Keywords: *Semantic Web, Link Open Data, Semantic Heterogeneity, Ontology Matching, Instance-based Matching, Training Set*

1. INTRODUCTION

Gradually, the importance of ontology based techniques in reading and processing semantic information is dominating the semantic web. Traditional search is being replaced by semantic search for effective presentations of query results to the user. For the information on the web to be shared and be interoperated, strategies to minimize if not totally eliminate the heterogeneities in the data became necessary. Even though, many techniques have been developed to solve semantic heterogeneity through ontology matching [1], [2] and [3] to mention a few. Yet, producing complete alignment remains competitive among many matching systems [4]. Most of these systems concentrate on the exploit of structures or schemas of the ontology while neglecting an important component of an ontology called instance or individual [5], [6]. Many approaches presented by different authors contribute significantly the naïve methods of matching ontology instances [7], [8] [9] and [10]. A recent survey on instance based matching reported that ontology matching will remain incomplete provided the ontology's instances are not match in an unsupervised fashion [11]. Even though, some instance matching systems recorded high F-score in generating alignments, still some aspects of the systems requires global attention.

Traditional instance matching method consists of two important steps (Figure 1), a blocking step and a similarity step [12]. A blocking involves grouping entities into paired and unpaired clusters, thereby making the paired group a candidate for a matching, this step can be replaced with machine learning clustering approach as in the work [13] where potential matching attributes are discovered using clustering method. A similarity step is characterized as classification stage where content of paired candidate group are classified and evaluated to produce the final output of the matching. In these systems, the specification function that specifies the mapping requires an

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

intervention from the domain expert to complete the matching process. In some approaches, the blocking has to consider manually trained examples for the mapping to take place. Therefore, these methods are completely supervised.

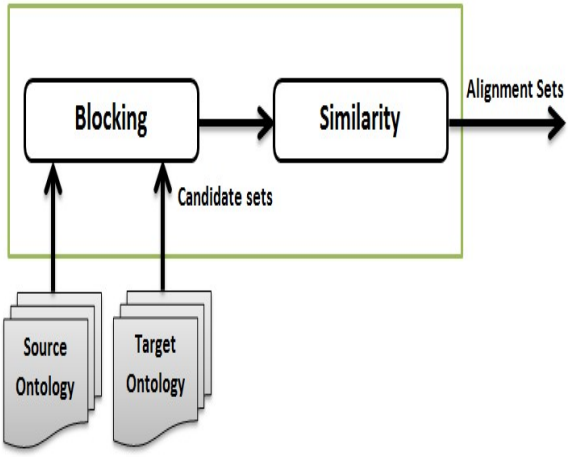

Figure 1: Traditional Instance-based matching Method

To address the supervision problem, [14] proposed an approach called training set generation to bootstrap the overall instance matching process. The idea suggested that as long as training set can be identified via suitable assumptions, the set would be appropriate to increase the adaptability of the whole matching process. However, this approach concentrated heavily on an assumption which does not often satisfy in most of the RDF graph data. In fact, this method is considered to be unsupervised and can only work effectively to domain-independence graph data. Most of the training set generation methods in the literature are experimentally weak when handling heterogeneous RDF data. [15], tries to bridge this gap in their work. In the work, they built a schema free instance matcher that aimed to be unsupervised but fails for some reasons. This system succeeded in matching RDF data that is made up of few properties but recorded low performance in mapping large-scale data. Therefore, designing and use of unsupervised training set generator in an instance matching system would address the above challenges.

Training set is the key role player that can improve the performance of any classification task. Different techniques and methods are being applied to generate training set depending on its area of application. Researchers in data science and semantic web community use different kind of training sets to improve the performances of classifications and information retrieval capability. In the work on instance matching, [16] uses supervised learning approach to generate a training set in order to improve the performance of their instance matcher. In a similar work, [17] generated a training set by constructing a novel hybrid method based on genetic algorithm for optimization. In [18], task-relevant training set was used in the object recognition task. It uses information that is in the language-based. In [15], generated training set is corporates for the first time to perform a property alignment in the input graph data. In this approach, semi-supervised approach is used to generate both negative and positive training sets to reduce too much comparison during matching. However, semi-supervised approach requires little human intervention in generating a training set, therefore, achieving automation will in one way or the other affect the matching process as a result of inferred function being produced for every new mapping. As irregular data is always part of the RDF graph data, the desired output in supervised and semi-supervised learning may contain noise due to the sensor or human errors, then the learning algorithm must be constrained to turn down the function that exactly maps the training samples. Therefore, we conducted this research to test the hypothesis that

Generating a training sets or samples in an unsupervised way rather than traditional supervised methods can effectively bootstrap the ontology instance matching performance.

In this paper, we design an unsupervised training set generator (UTSG) to produce training samples that can be fed to the property aligner in a complete unsupervised fashion. Avoiding human effort or sensor effort in generating a training sample will bootstrap the learning process, thereby making the whole process unsupervised. Thus, an important instance matching system requirement of automation can be guaranteed. In order to reject or fail to reject the hypothesis, this work addresses the following question: *will unsupervised training samples generation improve the performance of classification task with regard to ontology instance matching when compared with traditional approaches?*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1.1 Training Set Generation Intuition

Let introduce a real-world scenario to get insight on the training set generator, ponder two faculty members, *Hazlina Hamdan* and *Norwati Mustapha*. Consider these persons to be present in two different ontologies *O1* and *O2* that belong to a similar type (*person* and *people).* These two people must be proved by an effective instance matching system based on the equivalence relationship that exists in both ontologies. If the sets of these two person's information are represented as shown in Figure 2, for them to be declared equivalent with the approach that is solely *token-based* and ignored all other information (like that in property, phonetic and structure of the entities), the result of a mapping would be more precise (either similar or non-similar).

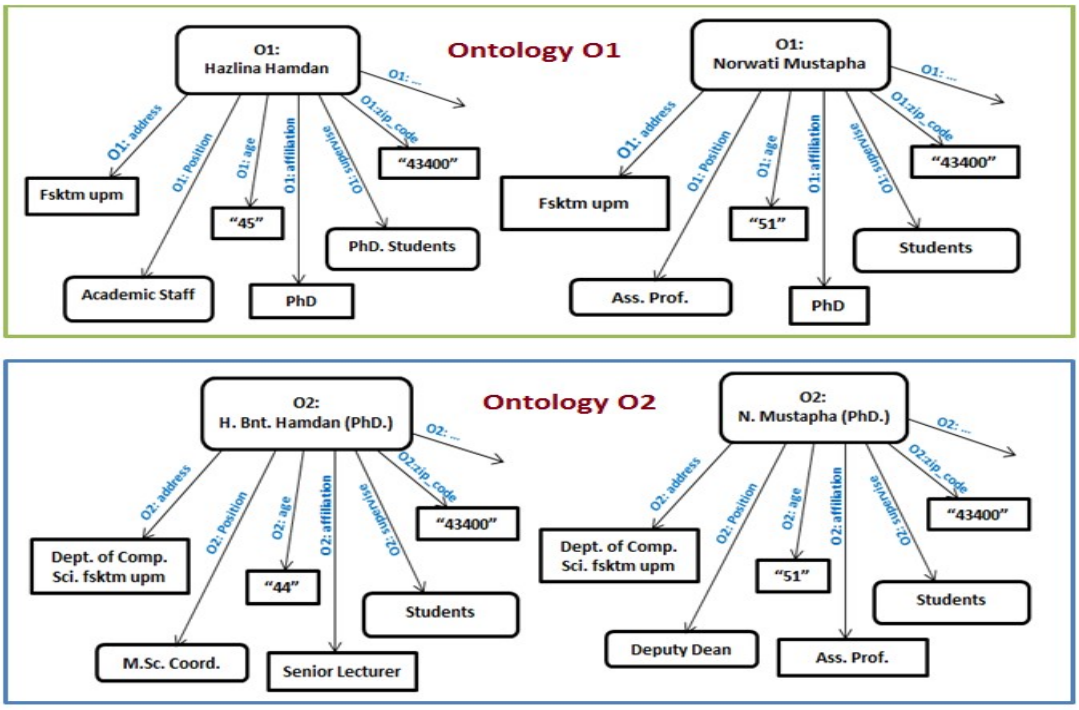

Figure 2: An Intuition that backs TSG Component

Let assume a token-based matching of these two different ontologies as mentioned earlier. In the information of an entities *Hazlina Hamdan and Norwati Mustapha*, cannot be declared the using equivalent relation as most of the attributes have different structure. The degree of similarity within the attributes may like be very low, especially when the mapping is strictly ***1:1.*** The in property (**:age)** for the entity *Hazlina Hamdan* in *O1* and that of *H. Bnt. Hamdan* in *O2* are completely different. Similarly in the property (***:position***) for the entity *Norwati Mustapha* where mapping between "Ass. Prof." and "Deputy Dean" might not yield a reasonable result. This intuition may be more extreme if we can assume that the name *Hazlina* typed wrongly in *ontology 1 as Hazrina* which is inevitable in records that encompasses manual data input. The change in only one letter may completely prove the mapping as non-equivalent. If we assume the generated set by the TSG returned "*Hazlina Hamdan"* as the training sample, many features can be learned from it as well as its corresponding entity. These feature set may include, token, numeric, phonetic and string features. In the case of misspelling of the name *Hazlina*, phonetic feature in the set may be of great advantage. Therefore, generating training samples can bootstrap the generation of alignment between ontologies property. Thus, the generated training set would be an input to the *property aligner* (next component in the workflow diagram).

Ontology integration is considered to be the general term used to describe different operations being conducted on the ontologies, such as features sharing, merging, unifying, mapping, aligning, and matching between different

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ontologies belonging to either same or different domain of the ontologies. Ontology integration is the process which may be done in three levels [19]:

i. Building a new ontology by reusing other available ontologies: this is the simplest case of ontology integration in which new ontology is built by adopting the existing ontologies.

ii. Merging different ontologies about the same domain into a single one that unifies all ontologies: In this case, the ontology should be built by using knowledge from exactly the same domain of existing ontologies.

iii. Introducing ontologies into the application: Here, several ontologies are introduced into an application, they are shared among different software applications which make it possible to use several ontologies to implement or identify knowledge-based applications on the basis of distributed resources.

On one hand, [20] identified two general approaches for ontology integration process: (1) Merging several ontologies for developing one consistent ontology, and (2) Alignment of several ontologies to be identifying their references to determine the possibility of employing all the ontologies. Therefore, to determine the inconsistency or conflict among ontologies one has to define and analyze several ontologies as suggested in case one above. In the second case, in every two candidate's ontology, it is required to find a mechanism which points out the relationship between the elements of both ontologies. In this case, it is possible to apply both ontologies for the set goal without unifying them to single ontology.

There are many pieces of evidence on why the one real-world entity is described in different sources. In the case of instance mentioned above, in open and social data, anyone has ample right to published data and/or information, and simply adhere to representation and that best fits his application. Another difference may be due to different data acquisition approaches such as the processing of scientific data. In addition, entities are dynamic in nature, they may evolve and change over time, and this development has to be up-to-date in data sources which are often either impossible or found to be hard enough (especially when this happens in a synchronous way). Finally, when integrating data from multiple sources, the process itself may add (new) erroneous data [9]

2. RELATED WORKS

Instance-based matching compares two or more sets of individuals of objects or classes so as to decide whether or not they can represent areal-world object figure 1.6. They combine items into a single form. Instance matching is an important aspect of ontology integration as it groups all important points of instances for better interoperability among different information sources [21].

There are many pieces of evidence on why one real-world entity is presented in different sources. In an open and social source of data, anyone has ample right to published data and/or information, and simply adheres to representation that suits his application. Another difference may be as a result of different data acquirement methods. Furthermore, entities are dynamic in nature; they change over time resulting in the frequent update which is often either impossible or found to be hard enough. Lastly, if data is being integrated from multiple sources, the integration process is bound to include noisy data [9]. In order to overcome such kind of problems, there is a global need for a standard benchmark for instance matching, for most instance matching techniques requires evaluation for them to suit the context of their applications. The benchmark will assist in determining the scope of existing techniques and identify the strength and weaknesses of these systems as well as support the advancement of instance matching research [9]. SEM+ implements a novel semantic similarity computation model called the Information Entropy and Weighted Similarity Model (IEWS Model) to suggest similarity measures between instances of distinct ontologies and vocabularies concepts [22]. Based on the similarity measures, SEM+ creates "same as" links among those concepts. SEM+ also implements a new prefix-based blocking algorithm, which groups possible matching pairs into one block. This blocking algorithm lessens the number of concept pairs that are needed for similarity calculation, which is important if mapping between two large domain ontologies is required

A. General Indexing Functions (GIFs) Used

In this work, we use 28 General Indexing Functions (GIFs) to construct our feature space. A GIF takes a string as input and returns a set of strings as output. We can think of a GIF as an 'atom' that we use to construct more complicated feature and hypothesis spaces. For the sake of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

completeness, we describe them in more detail here, complete with examples.

1. ***Identity:*** Returns a singleton set containing the string.

Example: The string "abcde" would simply be returned as {"abcde"}

2. ***Tokens:*** Tokenizes the string based on a set of delimiters specifically designed for RDF elements (see below), and outputs the set of tokens.

Example: Consider the string "7850 Avenue C; Apt. 103". The output would be {"7850", "Avenue", "C", "Apt.", "103"}. Note that the output depends strongly on a good tokenizer. In this example, it was assumed that whitespace and; are in the tokenizer set. The tokenizer that we uniformly use throughout the project is encoded by the Java statement, this tokenizer is also used in the training set generator.

3. ***Integers*****:** Similar to *Tokens* but discards all strings in the output that cannot be parsed as integers.

Example: Continuing from the previous example, this GIF would return {"7850", "103"}. Note that the output is always guaranteed to be a subset of the output of *Tokens*. The goal of such 'specialized' GIFs is to provide better discriminative ability for applicable cases.

4. ***ManipulateIntegersByOne:*** Same as *Integers*, except that for every integer *a*, integers *a-1* and *a+1* are converted to strings and added to the output set along with *a*.

Example: Again continuing from the previous example, this GIF would return {"7850", "7851", "7849", "103", "102", "104"}. Note that *Integers* is again a subset of this GIF, but this GIF is not necessarily a subset of *Tokens*, unlike *Integers*.

5-7. ***ExtractNCharPrefixes*****:** Same as *Tokens* except that each token is further truncated to its first N characters. If the token has fewer than N characters, it is left intact. Three GIFs were implemented, with N set to 3, 5 and 7 respectively.

Example: Continuing from the example started in *Tokens*, *Extract3CharPrefixes* yields {"785", "Ave", "C", "Apt", "103"}, *Extract5CharPrefixes* yields {"7850", "Avenu", "C", "Apt.", "103"} and *Extract7CharPrefixes* yields the same output as *Tokens*. Unless there are very long words in the string, the last observation almost always holds. In general, as N increases, the feature gains more discriminative ability. The choices of the odd numbers for N are arbitrary, but found in previous studies (and our own experiment) to work quite well.

8-10. ***ExtractTokenNGrams*****:** Tokenizes the string as an ordered list and extracts length-N contiguous subsequence of tokens. If the list of tokens contains fewer than N tokens, the list becomes its own only subsequence. Each subsequence is added to the output set. Implemented for N=2,4,6.

Example: Assuming the string "7850 Avenue C; Apt. 103" as input, the output for *ExtractToken2Grams* would be *{"7850 Avenue", "Avenue C", "C Apt.", "Apt. 103"}*, and similarly for N= 4,6. Notice how the delimiter; is not used, since the string is first tokenized and then converted to subsequence. Also, similar to GIFs 5-7, the choices of N are arbitrary but have been found to work well experimentally.

11-

17. ***ExtractNonSoundexPhoneticFeatures*****:** Tokenizes the string and adds the phonetic encoding of each token to the output set. The phonetic functions used for implementing seven GIFs in total are *Caverphone1* (Encodes a string into a Caverphone 1.0 value), *Caverphone2* (Encodes a string into a Caverphone 2.0 value), *ColognePhonetic I (*Encodes a string into a Cologne Phonetic value), *DoubleMetaphone* (Encodes a string into a double metaphone value), *MatchRatingApproachEncoder, Metaphone* and *NYSIIS*. A library implementing all these encoding functions efficiently exists in an Apache open-source package (https://www.apache.org/) and is adapted for this project.

18-

27. ***ExtractSoundexPhoneticFeatures*****:** Tokenizes the string and adds the *Soundex* encoding of each token to the output set. We consider the original Soundex encoding algorithm (implemented in the Apache open-source package), a refined version (also implemented in the package) as well as eight variations implemented in the open-source FEBRL package [14]. An example of a variation is to truncate each Soundex encoding to only the first four characters.

28. ***ExtractAlphaNumeric*****:** Extracts all tokens from the string such that a token contains at least one alphabet as well as a numerical digit (in addition to other optional characters). This GIF is used in the work of [23]

Example: Consider an input string "Sony Camera HD678941". The output of this GIF would be {"HD678941"}. Notice how the GIF concisely provides discriminative information when identifying strings are present. For example, if in another database, the same product was described in a slightly different manner (e.g. "Sony High-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Definition Camera, ID HD678941") the output of this GIF would be identical. Again, we note that the output is a subset of *Tokens.*

Most of these GIFs are empirically tested in the previous research. For example, GIFs 18 to 27, although the examples above show the utility and limitations of the GIFs, we have also provided a brief rationale for using them. An interesting area of future work is to conduct feature-specific research to ascertain how correlated each of these GIFs are with the output of an instance matching system. Such a study would obviously hang on more to the datasets and on the types of noise in the datasets. We believe that such a study would also help in gaining insight into the nature of the instance matching task, and why it's proving to be such a stubbornly difficult AI task.

3. PROPOSED UNSUPERVISED TRAINING SET GENERATOR (UTSG)

Achieving automation is one of the primary objectives of any instance matching system. The lower the level of supervision in the instance matching system the higher the level of automation the system can achieve. In this work, we aim to improve the efficiency of instance matching technique by introducing important technique to the traditional matching method, termed, Unsupervised Training Set Generator (UTSG), shown in Figure 3.

This technique is aimed to bootstrap the general matching process. The primary objective of UTSG is to provide input to the Property Aligner (PA) which is also a component to the traditional matching system. Both positive and negative training sets will be automatically generated with UTSG in linearly time frame. With UTSG, the training set to be produced are expected to contain minimum number of unpaired samples that can easy be accommodated by one the remaining components. Figure 3 describes the process flow of the proposed UTSG.

The pairs of record in D' list are measured with Jaccard similarity score [24]. If two bags of tokens t1 and t2 are given as input, their Jaccard similarity is defined as:

$$Jaccard[t1, t2] = \frac{t1 \cap t2}{t1 \cup t2} \quad (1)$$

One important characteristic of Jaccard similarity is that it is similarity function is local, meaning it depend not on external information set (like *IDF*) which requires searching over the entire dataset. As threshold and *Log Tf-Idf* are already applied to be a filter for removing unnecessary non-duplicate pairs, Jaccard score can serve to refine further the list of tokens and sort list *D*'. In fact, Jaccard similarity would eliminate many false-positive possibilities for being part of the list D. applying these two heuristics is of vital experimental benefit in filtering and sorting out tokens that can demonstrate high degree of overlap.

The constraint enforce in the algorithm is that each record from the property table appears at most once in a set D'. intuitively, the restrictions tries to make the unsupervised training set very much representative by preventing unnecessary records to appear in the training set. This is mostly a challenge of many existing *Training Set Generators* that are mainly semi-supervised such as in the work by [25].

Example 1: let n = 3 and the sorted list in D' in the algorithm is given as $Q' = [(i1, j3), (i2, j5), (i1, j7), (i6, j1)]$, where i and j represents the records *P1 and P2* in the property tables respectively. The matrix of the above list can be represented as:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

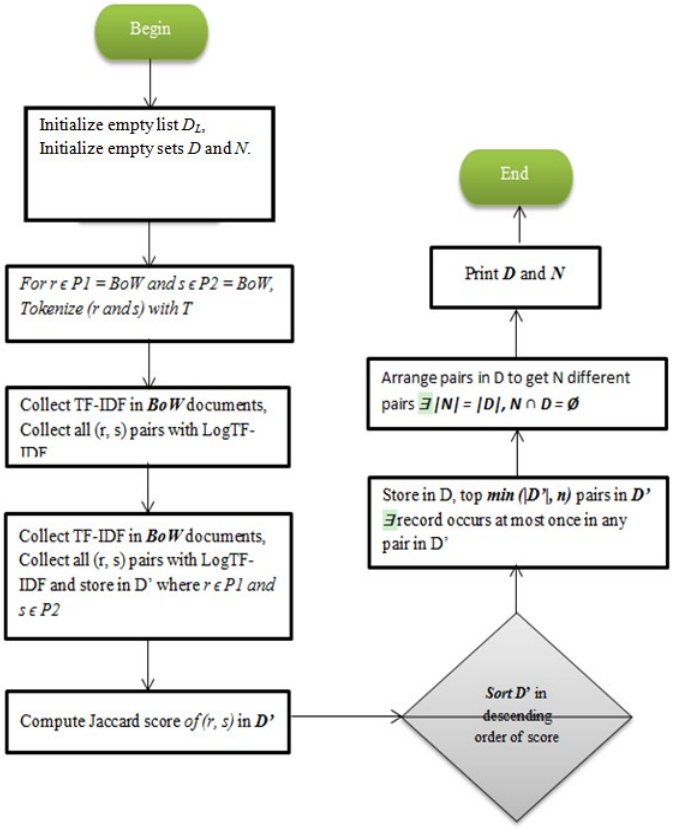

Figure 3: Utsg Process Flow

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
j_1						$i6, j1$	
j_2							
j_3	$i1, j3$						
j_4							
j_5		$i2, j5$					
j_6							
j_7	$i1, j7$						

With $n = 3$, the positive training set will be $D = [(i1, j3), (i2, j5), (i6, j1)]$, since the record i1 appeared most in the scoring pairs.

Example 2: proceeding from example 1, the generated set $D = [(i1, j3), (i2, j5), (i6, j1)]$, the possible non-duplicates set *G* by permuting *D* may be derived from the table below:

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
j_1	$i1, j1$	$i2, j1$					
j_2							
j_3		$i2, j3$					
j_4							
j_5	$i1, j5$						
j_6					$i6, j5$		
j_7							

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The possible set $D = [(i1, j1), (i2, j3), (i6, j5)]$, in practice these permutation yields a near perfect result in terms of accuracy on the generated sets.

3.1 Evaluation

Goal: To evaluate the *domain-independence and scalability* empirically using our proposed Unsupervised Training Set Generator (UTSG). We conducted evaluation on nine cross-domain test cases that cover about twenty data types with some of them multi-type.

The statistics of nine paired test suites used in the evaluation of UTSG are summarized in Table 1. In all nine data contains pair of distinct serialized files. All data sets are real-world benchmarks made available via ontology matching and semantic web competitions (OAEI[1]). Almost, all the test cases share the same type information. Therefore, the type alignment problem has been taken care off in the data sets, so we do not consider it here. This is why the type alignment is excluded in our matching process flow diagram. This is a right benefit of using real-world benchmark data sets in an evaluation otherwise another module for type alignment has to be constructed in order to generate a reliable alignment. Our analysis criteria consists of the following parameters (data type, data size, data distribution, parameter thresh, number of instances and data transformation). The data transformation is characterized by data structure, data type and data semantics [26]. The important factors are the number of instances pairs as used in many previous training set algorithms such as the work of [27] to be compared among the instances pairs with a very small parameter thresh (thresh ≠ 0). These factors are used to measure the conformance of the generated training samples with maximum precision and recall in which every potential attribute can be considered as a training sample. These training samples can be used to generate matching between the semantically related objects. The data distribution for the synthetic data used ranges from 0.0 to 1.0 (0.2, 0.4, 0.6, 0.8 and 1.0) as shown in figures 4, 5 and 6.

Table 1: Test Suites Statistics

Matching task	Ontology's classes	Number of Pairs	Total Instances
Sanbox003	owl: NamedIndividual owl: NamedIndividual	363 367	133221
Person 1	Person_11: Person Person_12: Person	2000 1000	2000000
Person 2	Person_21: Person Person_22: Person	2400 800	1920000
IM_Identity	identity_a: Book identity_b: Book	1330 2649	3523170
IM_Similarity	Similarity_a:Book Similarity_b:Book	1675 1658	2777150
IIMB_005	Film:Science_fiction Film:Science_fiction	581 222	128982
IIMB_010	Film:Science_fiction Film:Science_fiction	2150 1568	3371200
IIMB_015	Film:Science_fiction Film:Science_fiction	8441 1416	11952456
SABINE	Source: Topic Target: Topic	706 1127	795662

3.2 Measurement Metrics

We apply commonly used measurement metrics, precision and recall as well as their harmonic mean (F-Measure) as the evaluation metrics. Precision is the fraction of relevant instances among the retrieved instances while Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are based on the measure of relevance. The ground truth in this experiment is the set of true positives in both our UTSG and the baseline TSGs. In this context the metrics are formally defined as:

$$Precision = \frac{True_{Positive}}{True_{Positive} + False_{Positive}} \quad (2)$$

$$Recall = \frac{True_{Positive}}{True_{Positive} + False_{Negative}} \quad (3)$$

In real-world scenarios, the trade of between efficiency and effectiveness is frequently observed. Therefore, F-Measure is applied to represent this trade-off. It is used to approximate the average of precision and recall value. F-Measure can be defined mathematically as follows:

[1] Ontology Alignment Evaluation Initiative organizes annual campaign in order to evaluate ontology matching systems.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$F - Measure = 2 . \frac{Precision . Recall}{Precision + Recall} \quad (4)$$

3.3 Experimental Set-up

To the best of our knowledge, DUMAS[2] TSG [16] and a semi-supervised TSG proposed by [15] are the only existing works that detect matches in structurally heterogeneous data sets. Thus, we use them as the baseline in this experiment. Both approaches applied *LogTfIdf* to point the required duplicate set (process 2-3 in our flow chart, (fig.3)). In the first action, precision-recall trade-off is plotted by applying the *Jaccard similarity* score (process step 4 in the flow chart, (fig.3). We ran the paired t-test to measure the statistical significance by comparing F-Scores generated by our system against each of the baseline system. The parameter threshold is asset as 0.02 in the algorithm and it is sets to be in a *self-tuning* mode. *Self-tuning* mode will make the *threshold* acceptable for the variety of test suites.

We used large number of instances (*n* value) in our experiment so that the underlying properties of the data sets can be represented. About **700** instances are considered as *n* value of our algorithm. With this value of *n*, the algorithm is expected to eliminate all imperfectly classified pairs of instances to be present in the training set. In selecting these instances, the elements or attributes have to undergo series of sorting and re-sorting so that all selected instances appeared once in the list (sort step in the flow diagram). Finally, the selected non-duplicate is rearranged and the results are recorded base on the precision, recall and F-score measures of 758 instances. For better justification, this procedure is repeated for our baseline TSGs techniques. The report of the experiment is presented and discussed in the subsequent sections. All experiments were conducted in *Python* programming environment with statistical test carried out in *R* programming language in order to obtain a reliable result.

3.4 Results and Discussions

Table 2 shows the results of our proposed UTSG against the baseline TSGs: KEJ_TSG and DUMAS_TSG. The result is recorded according to the highest obtained F-Measure either greater or equal to 80% in both precisions and recalls for all test cases. Our proposed method outperforms all baseline TSGs in both precision and recall which also resulted in having high F-measure in our approach as against the baseline TSGs. The poor performance with average F-Measure of 58% obtained by DUMAS_TSG in the experiment clearly indicates that supervised TSG is inappropriate for bootstrapping the matching and RDF data linkage. This is because the difference of 35% between the F-Measure of our UTSG and that of DUMAS TSG is too significant. On the other hand, KEJ_TSG also performs considerability better than DUMAS TSG. The F-Measure obtained by KEJ_TSG is almost high than that obtained by DUMAS_TSG in the entire test cases, except in *IIMB_010* and *Sabine* data sets.

Despite the average F-Measure recorded by KEJ_TSG which is below our benchmark of 80%, we can still conclude that KEJ-TSG can fairly bootstrap the matching process in contrast to DUMAS_TSG. By extension, one can say that semi-supervised TSG is suitable for bootstrapping the matching and RDF data interlink than the complete supervised TSG. In contrast to KEJ_TSG, our proposed complete unsupervised method performs significantly better with the average F-Measure of 93% as against 76% recorded in KEJ_TSG baseline. In the end, the hypothesis holds with statistically significant difference at (P-value < 0.05).This empirical results show that developing TSG in complete unsupervised fashion will yield to a better performed ontology matching system. Even though, execution time not considered in this experiment but careful observation of the running times of both our proposed method and the baseline TSGs demonstrated competiveness in all the test cases and performed near linear. However, the general running time will be of great importance during the final run of the system (that is when all sub-components under construction in our ongoing project are integrated to generate final and complete alignments).

[2] Duplicate-based Matching of Schemas

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

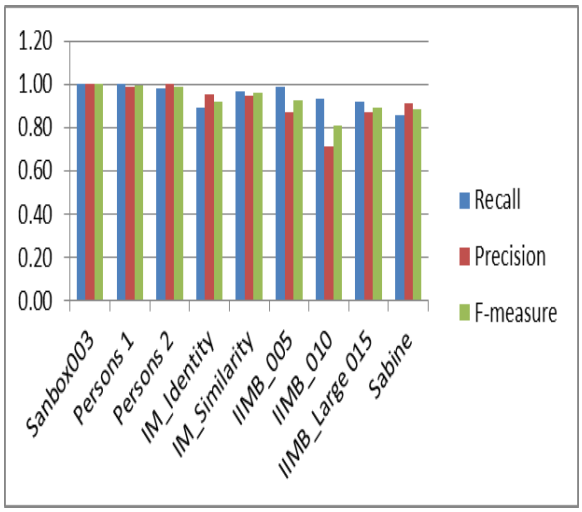

Figure 4: Results showing the Performance of Proposed UTSG obtained in all Nine Test Cases

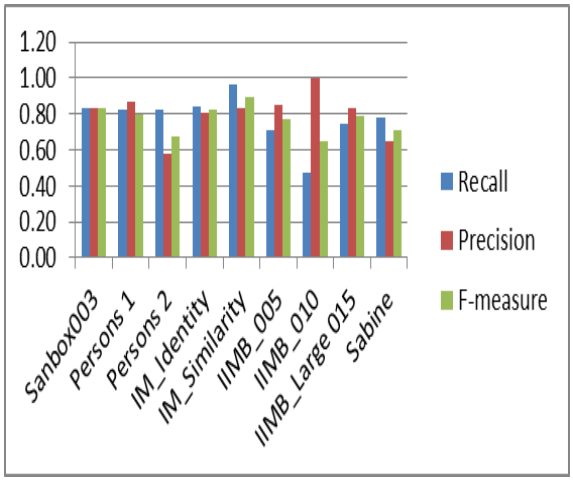

Figure 5: Results showing the Performance of Kejriwal TSG obtained in all Nine Test Cases

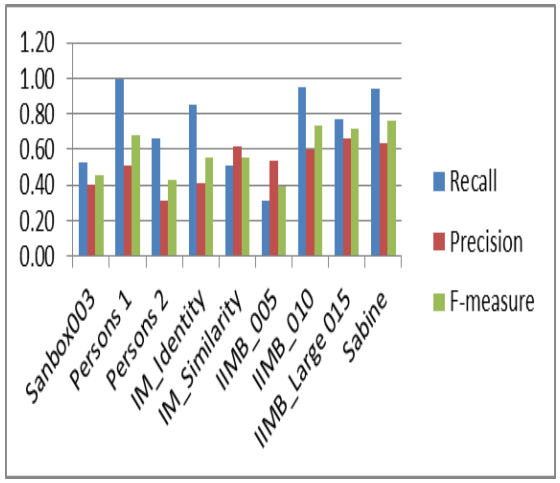

Figure 6: Results showing the Performance of Dumas TSG obtained in all Nine Test Cases

In Figure 4, 5 and 6, the recall, precision and F-measure obtained in all the test cases by applying proposed UTSG is graphically represented. Except in *IM-Identity* and *Sabine* test cases, all test cases achieved the maximum recall more than 90% which signifies the presence of high true-positives in the generated sets. This indicates that our proposed UTSG yields a positive result. The fall in recall below 90% experienced by IM-Identity and Sabine arose as a result of too much irregular data found in the two data sets and this is one of the important issues that the remaining components would take care off to improve the efficiency of the system in producing final output. This result clearly shows that our UTSG can also perform well in a cross domain scenarios as the nine test cases used for the experiment came from different domains. Furthermore, UTSG also addresses another drawback of most instance matchers for being non-scalable systems. Regardless of the size of the ontologies, our UTSG effectively generated training samples in all the nine test cases in a near linear running time. This is why the running time is not a priority in this experiment but rather in the final output of the system.

Even though, the semi supervised TSG proposed by Kejriwal (Figure 5) performs reasonably better than DUMAS TSG (Figure 6), both experience some falls in both recall and precision in many of the test cases. However, their performances are still low compared to that of UTSG. These falls led to a trade-off by having low F-Measures in the baseline TSGs compared to our proposed UTSG (Table 2). The average F-Measures (Figure 7) shows that our approach with 93% F-Score outperforms the two baseline TSGs (KEJ TSG and DUMAS TSG) with F-Score 76% and 58% respectively. This result also shows the suitability of our approach in addressing scalability and controlling the trade-off between effectiveness and efficiency of the matching systems. Yet, all the three TSGs demonstrated the ability of being Domain-independence. To the base of our knowledge, these baseline TSGs are the only learning-based TSGs found to be applicable in bootstrapping the RDF data matching and populating linked data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Comparative Analysis for the Proposed UTSG with Baseline TSGs

	Proposed UTSG			KEJ_TSG			DUMAS_TSG		
Test Suits	**Recall**	**Precision**	**F-Measure**	**Recall**	**Precision**	**F-Measure**	**Recall**	**Precision**	**F-Measure**
Sanbox003	1.00	1.00	1.00	0.83	0.83	0.83	**0.53**	**0.40**	0.46
Persons 1	1.00	0.99	0.99	0.82	0.87	0.80	0.99	**0.51**	0.68
Persons 2	0.98	1.00	0.99	0.83	**0.57**	0.68	**0.67**	**0.32**	0.43
IM_Identity	0.89	0.95	0.92	0.84	0.81	0.82	0.85	**0.42**	0.56
IM_Similarity	0.97	0.95	0.96	0.96	0.83	0.89	**0.51**	**0.62**	0.56
IIMB_005	0.99	0.87	0.93	**0.71**	0.85	0.77	**0.31**	**0.53**	0.39
IIMB_010	0.93	**0.71**	0.81	**0.48**	1.00	0.65	0.95	**0.60**	0.73
IIMB_ 015	0.92	0.87	0.89	**0.74**	0.84	0.79	**0.77**	**0.66**	0.71
Sabine	0.86	0.91	0.88	**0.78**	**0.65**	0.71	0.94	**0.64**	0.76
AVERAGE	**0.95**	**0.92**	**0.93**	**0.77**	**0.80**	**0.76**	**0.72**	**0.52**	**0.58**

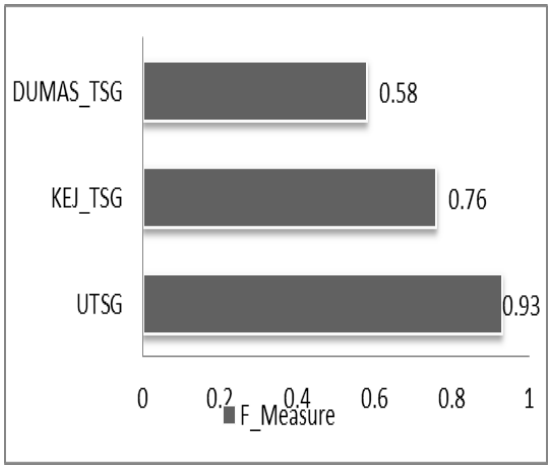

Figure 7: Average F-Measures of UTSG with Baseline TSGs obtained from all Nine Test Cases

Traditional instance-based matching techniques analyze candidate attributes separately; they extract attribute's properties like average length of character, average length of strings, and ratio of the attribute length and so on. Attributes with corresponding properties are always assumed to be the same in meaning. This kind of approach is normally called, vertical matching as a result of comparing properties of table columns. In DUMAS method [16], horizontal matching is performed. They traverse tables to search similar tuples (rows) to detect duplicates. Their horizontal approach solves two important matching problems: (i) fuzzy duplicates detection and (2) generating a schema matching using the duplicates set. In this method, duplicates can be found even if the overlap is small. In [15], TSG is designed to tokenize each row (tuple) in a property table by applying a tokenizer, then transforms it into bag-of-words through a standard information retrieval method *Term Frequencies and Inverse Document frequencies (TF-IDF).* To the best of our knowledge, TSGs are applied to record linkage problem and they are considered to work in a supervised and semi-supervised fashion respectively and demonstrated low performance compared to our approach.

All four types of validity threats have been monitored with due consideration to the type of research in prioritization. In this work, the priority for experiment is in the order: Internal, external, construct, and conclusion. In conducting this experiment, we identified and handled two important threats to validity: internal and external validity. Before the commencement of the experiment, we ensure that the experimental environment is appropriate to carry out the experiment. The amount of memory needed and CPU speed are ensured to be in conformity with the baseline methods used in this work. Thus, threat to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

internal validity has taken care of. To ensure generalization of this work and easy replication in different environment, we tested our algorithm with nine benchmark datasets which are publically available in OAEI website which is cross-domain. The technique is also compared with two state-of-the-art techniques of training set generation. Thus, threat to external validity addressed.

4. CONCLUSION

Linked Data success is the important aspect for web applications success. Most importantly, if the data to be linked across two or more RDF graph, or to be replicated in the same source of data (i.e. self-match) may link to different domains, such as health, publication, people, production, agriculture, etc. for instance, with the growing quantity of online shopping websites, instance matching is the best technology that provides accurate and all-inclusive price unification among similar products by linking all instances of related items together and presents it to user as required. A shopping site (such as AMAZON) can be able to precisely identify similar products alongside their prices with the power of semantic interoperability and record matching.

For web search applications (named, search engines), instance matching is an important technology that facilitates the removal of duplicate query results. Moreover, instance matching technique is also broadly applied to knowledge management systems that provide a mechanism to drive new knowledge through integrating similar data when preparing for data mining and statistical analysis.

The significance of instance matching gains more relevance if one consider that lot of real-world object (such as, place, people, event) are being represented on the web within many documents in a heterogeneous form of representation of individuals or instances. Furthermore, instance-based matching is highly needed in the areas of ontology management as it tremendously assists domain experts in performing ontology manipulations via advancement and improvements in the ontology engineering techniques and models. For example, instance-based matching boosts *ontology population* (i.e., make a new inclusion of an individual in an ontology) and also to identify the similarity possibility to map incoming instance with the existing ones.

The output of our UTSG shows that, the implementation of the algorithm can successfully take care of data irregularity (heterogeneity) which is a necessary concern in different problems that are data-intensive. It is observed that with efficient UTSG, it is possible to have a complete unsupervised instance matching system that could not compromise quality. This will allowed us to achieve the desired aim of developing an instance matching system that can satisfy the scalability and automation requirements despite the heterogeneous nature of the RDF graph data. However, discovering when the UTSG will be selected automatically over a self-adaptive classifier is a serious limitation of this work which is open for further research.

There are many direction of work intended to undertake in a near future. This includes cloud implementation of this method as MapReduce to ensure real time scalability test.

ACKNOWLEDGMENT

This work is supported by University Putra Malaysia grant (Putra Grant No: 9569200) and Al-Qalam University Katsina (AUK), Katsina State Nigeria under the University's Staff Development Unit.

REFERENCES

[1] J. Euzenat and P. Shvaiko, *Ontology matching*, vol. 18. 2007.

[2] L. Liu, F. Yang, P. Zhang, J.-Y. Wu, and L. Hu, "SVM-based ontology matching approach," *Int. J. Autom. Comput.*, vol. 9, no. 3, pp. 306–314, 2012.

[3] I. Akbari and M. Fathian, "A novel algorithm for ontology matching," *J. Inf. Sci.*, vol. 36, no. 3, pp. 324–334, 2010.

[4] E. Jim´enez-Ruiz, "Results of the Ontology Alignment Evaluation Initiative 2014," *Proc. 8th ISWC Work. Ontol. matching*, pp. 61–100, 2013.

[5] M. Chi, Z. Yao, and S. Liu, "A Matching Algorithm Based on Association Rules in Ontology Based Publish/Subscribe System," *Chinese J. Electron.*, vol. 24, no. 1, pp. 65–70, 2015.

[6] A. Isaac, L. van der Meij, S. Schlobach, and S. Wang, "An empirical study of instance-based ontology matching," *Belgian/Netherlands Artif. Intell. Conf.*, pp. 317–318, 2008.

[7] S. Castano, A. Ferrara, D. Lorusso, and S. Montanelli, "On the ontology instance matching problem," in *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, 2008,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

pp. 180–184.

[8] J. Li, Z. Wang, X. Zhang, and J. Tang, "Large scale instance matching via multiple indexes and candidate selection," *Knowledge-Based Syst.*, vol. 50, pp. 112–120, 2013.

[9] E. Daskalaki, G. Flouris, I. Fundulaki, and T. Saveta, "Instance matching benchmarks in the era of Linked Data," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 39, pp. 1–14, 2016.

[10] S. Cerón-Figueroa, I. López-Yáñez, W. Alhalabi, O. Camacho-Nieto, Y. Villuendas-Rey, M. Aldape-Pérez, and C. Yáñez-Márquez, "Instance-based ontology matching for e-learning material using an associative pattern classifier," *Comput. Human Behav.*, vol. 69, pp. 218–225, 2017.

[11] Teh Noranis Mohd Aris, Mansir Abubakar, Hazlina Hamdan, Norwati Mustapha, "Instance-Based Ontology Matching : A Literature Review," in *Recent Advances on Soft Computing and Data Mining, Advances in Intelligent Systems and Computing*, 2018, pp. 456–469.

[12] S. Castano, A. Ferrara, S. Montanelli, and G. Varese, "Ontology and instance matching," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6050, pp. 167–195, 2011.

[13] M. Abubakar, H. Hamdan, N. Mustapha, T. Noranis, and M. Aris, "Attributes Correspondence Discovery in Ontology Instance-based Matching and RDF Data Linkage using Clustering Method," *Int. J. Eng. Technol.*, vol. 7, pp. 290–297, 2018.

[14] D. Vatsalan, P. Christen, and V. S. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *Inf. Syst.*, vol. 38, no. 6, pp. 946–969, 2013.

[15] M. Kejriwal and D. P. Miranker, "An unsupervised instance matcher for schema-free RDF data," *J. Web Semant.*, vol. 35, pp. 102–123, 2015.

[16] A. Bilke and F. Naumann, "Schema Matching Using Duplicates," *21st Int. Conf. Data Eng.*, pp. 69–80, 2005.

[17] N. Nahapetian, M. Analoui, and M. R. Jahed Motlagh, "Training set generation using fuzzy logic and dynamic chromosome based genetic algorithms for plant identifiers," *2009 IEEE Symp. Comput. Intell. Control Autom. CICA 2009 - Proc.*, pp. 49–56, 2009.

[18] M. Schoeler, F. Worgotter, M. J. Aein, and T. Kulvicius, "Automated generation of training sets for object recognition in robotic applications," *23rd Int. Conf. Robot. Alpe-Adria-Danube Reg. IEEE RAAD 2014 - Conf. Proc.*, 2015.

[19] H. S. Pinto and J. P. Martins, "Ontology Integrations: How to perform the Process," *Work. Ontol. Inf. Shar. (into IJCAI'2001)*, pp. 71–80, 2001.

[20] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Stanford Knowl. Syst. Lab.*, p. 25, 2001.

[21] R. P. Deb Nath, H. Seddiqui, and M. Aono, "Resolving scalability issue to ontology instance matching in Semantic Web," *Proceeding 15th Int. Conf. Comput. Inf. Technol. ICCIT 2012*, pp. 396–404, 2012.

[22] L. Li, X. Xing, H. Xia, and X. Huang, "Entropy-Weighted instance matching between different sourcing points of interest," *Entropy*, vol. 18, no. 2, pp. 1–15, 2016.

[23] S. Sowmya Kamath, V. S. Ananthanarayana, Sowmya Kamath S., and Ananthanarayana V.S., "Similarity analysis of service descriptions for efficient Web service discovery," *Data Sci. Adv. Anal. (DSAA), 2014 Int. Conf.*, pp. 142–148, 2014.

[24] F. Cao, J. Z. Huang, and J. Liang, "A fuzzy SV-k-modes algorithm for clustering categorical data with set-valued attributes," *Appl. Math. Comput.*, vol. 295, pp. 1–15, 2017.

[25] M. Kejriwal and D. P. Miranker, "Semi-supervised instance matching using boosted classifiers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9088, pp. 388–402.

[26] M. del C. Legaz-García, M. Menárguez-Tortosa, J. T. omás Fernández-Breis, C. G. Chute, and C. Tao, "Transformation of standardized clinical models based on OWL technologies: from CEM to OpenEHR archetypes," *J. Am. Med. Inform. Assoc.*, vol. 22, no. 3, pp. 536–544, 2015.

[27] A. Judea and H. Sch, "Unsupervised Training Set Generation for Automatic Acquisition of Technical Terminology in Patents Br ¨," pp. 290–300, 2014.