B.Sc. in Computer Science and Engineering Thesis

Inter Word Semantic and String Distance Using Ontological Techniques

Submitted by

Sanjid Habib Sultan
Id: 201114034

Afrin Jahan Noumen
Id: 201114039

Lamia Alam
Id: 201114053

Supervised by

Dr. Muhammad Masroor Ali

Professor

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka-1205, Bangladesh

Department of Computer Science and Engineering
Military Institute of Science and Technology

December 2014

CERTIFICATION

This thesis paper titled **"Inter Word Semantic and String Distance Using Ontological Techniques"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December 2014.

Group Members:

Sanjid Habib Sultan

Afrin Jahan Noumen

Lamia Alam

Supervisor:

Dr. Muhammad Masroor Ali
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1205, Bangladesh

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis paper, titled, "Inter Word Semantic and String Distance Using Ontological Techniques", is the outcome of the investigation and research carried out by the following students under the supervision of Dr. Muhammad Masroor Ali, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1205, Bangladesh.

It is also declared that neither this thesis paper nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Sanjid Habib Sultan
Id: 201114034

Afrin Jahan Noumen
Id: 201114039

Lamia Alam
Id: 201114053

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor, Dr. Muhammad Masroor Ali, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1205, Bangladesh, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Dhaka
December 2014

Sanjid Habib Sultan

Afrin Jahan Noumen

Lamia Alam

ABSTRACT

Ontologies are today a key part of every knowledge based system. They provide a source of shared and precisely defined terms, resulting in system interoperability by knowledge sharing and reuse. Unfortunately, the variety of ways that a domain can be conceptualized results in the creation of different ontologies with contradicting or overlapping parts. For this reason ontologies need to be brought into mutual agreement (aligned). Thus ontology matching between words is a useful technique for data integration and data sharing. Two important methods for ontology matching is the comparison of words using semantic similarity and string distance metrics. In our thesis work, we have used a lexical database called WordNet to find semantic similarity between words. String metric based similarity has been found out by using Jaro Winkler distance on the basis of commonality and differences between two words.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

COMA : Composite Schema Matching System
Idk : I don't know
OAEI : Ontology Alignment Evaluation Initiative

LIST OF SYMBOLS

$\equiv$: Equivalence relation
$\supseteq$: Superset relation
$\subseteq$: Subset relation
$\perp$: Disjointness relation
$\exists$: existential quantification
$\wedge$: Logical AND relation
$\in$: Set membership

CHAPTER 1
INTRODUCTION

1.1 General Overview

Our thesis topic is Inter Word Semantic and String Distance Using Ontological Techniques. To begin with an ontology typically provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. Depending on the precision of this specification, the notion of ontology encompasses several data/conceptual models, for example, classifications, database schema, fully axiomatized theories. Ontology matching [1] is a key interoperability enabler for the Semantic Web, as well as a useful tactic in some classical data integration tasks dealing with the semantic heterogeneity problem. To build a collaborative semantic web, which allows data to be shared and reused across applications, enterprises, and community boundaries, it is necessary to find ways to compare, match and integrate various ontologies. Different strategies (e.g., string similarity, synonyms, structure similarity and based on instances) for determining similarity between entities are used in current ontology matching systems. Synonyms can help to solve the problem of using different terms in the ontologies for the same concept. WordNet [2] is a lexical database for the English language. It groups English words into sets of synonyms called synsets, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members. WordNet can thus be seen as a combination of dictionary and thesaurus. The WordNet can support improving similarity measures. So it is widely used for measuring similarities. Our thesis work relates with the ontology matching. This paper emphasizes on String Matching and Semantic Matching of two words. We provide our findings about the similarities and dissimilarities of both the words using both the matching techniques and provide with our analysis in this paper.

1.2 Preview on Methodology

We wanted to find the similarities of two words using ontology matching. To achieve that we used two techniques Semantic Matching and String Matching.

We achieve the similarities of two words semantically by implementing WordNet based equality. It compares the two words with WordNet database and results in a number between

[0-1]. This represents the semantic relation of the given words.

String matching is another similarity measures we are conducting. This is based on both common and uncommon substrings in the two words. After finding their common and uncommon entities we cannot get the best results. So to improve our results we add Jaro Winkler Distance of both the words with our current result. This provides us with a better result on similarities.

We analyze our results with other similarity matchings to compare the similarities and dissimilarities of two words.

1.3 A Glance at the Chapters

Chapter 2 discusses about the background study on our work topics. Chapter 3 discusses about Semantic Matching and its usage in our work. Chapter 4 discusses about the string matching on our work topics. Chapter 5 discusses about the methodology of our work. Chapter 6 analysis of our thesis results. Chapter 7 contains the overall discussion and observation of our thesis.

CHAPTER 2
BACKGROUND STUDY

This chapter includes the background study of our thesis related topics.

2.1 Ontology

By an ontology [1] we mean here a way of defining a conceptualization of an application domain in terms of concepts, attributes, and relations expressed in a formal language. Relations can be defined by the user, but there are some pre-defined relationships with known semantics, i.e., is-a; part-of; instance-of. A concept hierarchy is an ontology without attributes and only with is-a relations between elements.

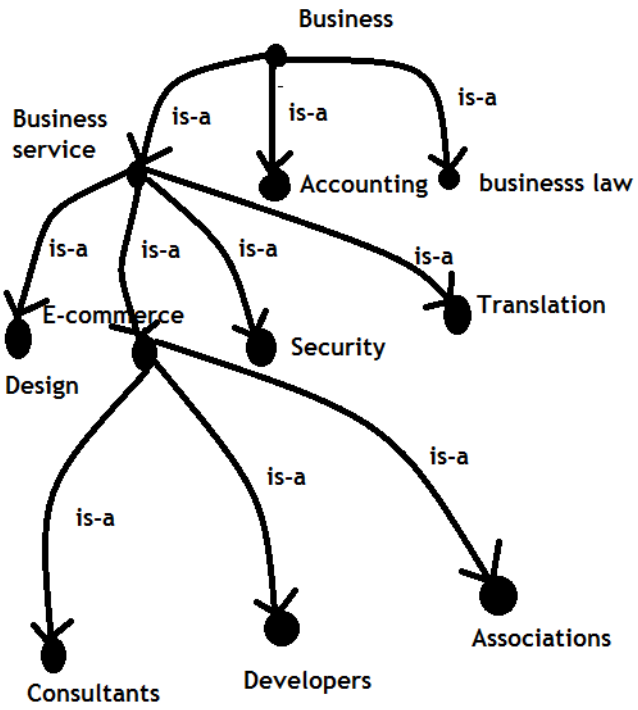

Figure 2.1: Google web directory.

One example of ontology can be constructed by complicating the concept hierarchy shown

in Figure 2.1, by adding attributes to the concept Association, see Figure 2.2. Attributes of the concept Associations are BN, City, Street, Zip, while data instances are B8 and B2. Data instances have fixed attributes values: instance B8 has BN="B8", City="Trento", Street="Piazza Venezia", etc.

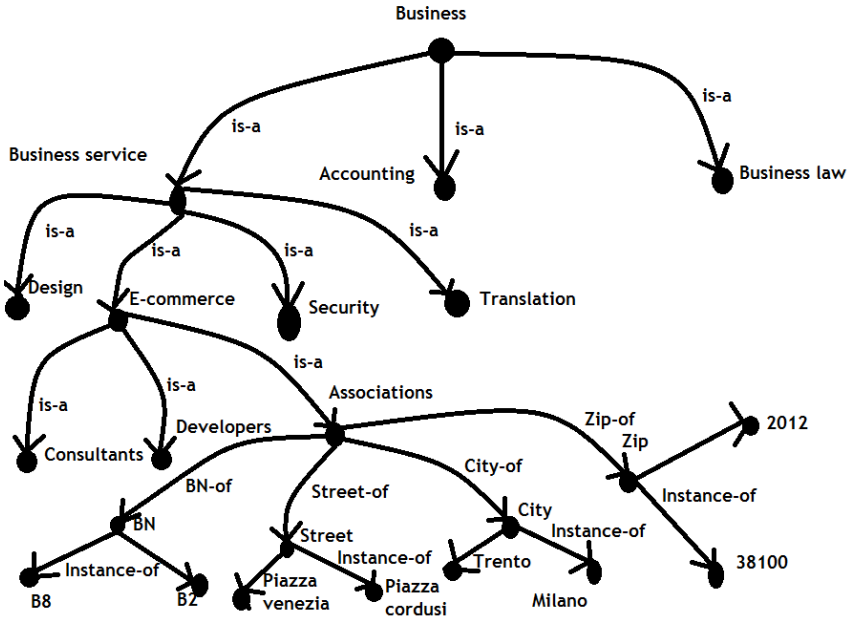

Figure 2.2: Example of ontology "Business".

2.2 Ontology Matching

Matching is a critical operation in many application domains, such as semantic web, schema/ontology integration, data warehouses, e-commerce, query meditation, etc. It takes as two input schema/ontologies, each consisting of a set of discrete entities (e.g. tables, XML elements, classes, properties, rules, predicates) and determines as output the relationships (e.g. equivalence, subsumption) holding between these entities. The ontology matching problem may be considered as a problem of matching the hierarchies of concepts and relations of two ontologies. Without the hierarchy of relations , an ontology can be regarded as a schema. In the schema matching domain, there was a lot of researches in the context of data integration and data translation. A good survey of approaches to automatic schema matching is presented in [3].

Despite its pervasiveness, today ontology matching is largely conducted by hand, in a labor-intensive and error-prone process. The manual matching has now become a key bottleneck in building large scale information management systems. The advent of technologies such as the WWW, XML and the emerging Semantic Web will further information sharing applications and exacerbate the problem. Hence, the development of tools to assist in the ontol-

ogy matching process has become crucial for the success for a wide variety of information management applications.

There is not very much work in the ontology matching domain. Some of recent researches are GLUE [4], Anchor-PROMPT [5] and SAT [6].

GLUE [4] is the evolved version of LSD [4] whose goal is to semi automatically find schema mappings for data integration. Like LSD, GLUE use machine learning techniques to find mappings. In GLUE, there are several learners, which are trained by data instances of ontologies. After learning phase, different characteristic instance patterns and matching rules for single elements of the target schema are discovered. The predictions of individual matchers are combined by a meta-learner and from that, final mappings result will be deduced.

Anchor-PROMPT [5] constructs a directed labeled graph representing the ontology from the hierarchy of concepts (called classes in the algorithm) and the hierarchy of relations (called slots in the algorithm), where nodes in the graph are concepts and arcs are the relationships between concepts (the labels are the names of the relations). An initial list of anchors-pairs of related concepts defined by the users or automatically identified by lexical matching is the input for the algorithm. Anchor-PROMPT analyzes then the paths in the sub-graph limited by the anchors and it determines which concepts frequently appear in similar positions on similar paths. Basing on these frequencies, the algorithm decides if these concepts are semantically similar concepts.

SAT [1] takes as input two graphs of concepts and produces as output relationships such as equivalence, overlapping, mismatch, more general and more specific between two concepts. The main idea of this approach is to use logic to encode the concept of a node in the graph and to use SAT to find relationships. The concept at a node, which is then transformed into a propositional formula, is the conjunction of all the concepts of the nodes on the path from the root of the graph to that node. The concept of a node, in turn, is extracted from WordNet. WordNet is a lexical database. The relationship which needs to be proved between two concepts is also converted to a propositional formula. SAT solver will run on the set of calculated propositional formulas to verify if the assumed relationship is true or not.

Initially, ontologies were used in an attempt to dene all the concepts within a specic domain and their relationships. An example of a popular ontology is WordNet [2], which models the lexical knowledge of a native English speaker.

Our goal is to apply techniques for ontological matching similarity measures, semantic comparison and string comparison (e.g. JaroWinkler metric). We do this for calculating the similarity value between descriptions of the concepts or the relations between two words. We also integrate the WordNet lexical database.

2.3 WordNet

WordNet [2] is an external terminological dictionary and a lexical database for English language which basically includes noun, verb, adverb and adjective lexical catagories. WordNet was created in the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George Armitage Miller starting in 1985 and has been directed in recent years by Christiane Fellbaum. The project received funding from government agencies including the National Science Foundation, DARPA, the Disruptive Technology Office (formerly the Advanced Research and Development Activity) and REFLEX. George Miller and Christiane Fellbaum were awarded the 2006 Antonio Zampolli Prize for their work with WordNet.

Description based similarity measure is often called terminological similarity measure as a description usually contains informative issues. The reason behind using WordNet in determining similarity measure is, it groups English words into sets of synonyms called synsets. Information in WordNet is organized around lexical groupings called synsets and semantic pointers. Informally, a synset represents a set of synonym words, while a semantic pointer models the semantic relationships between two synsets. Words like "benefit" and "profit" are synonyms which are grouped in Wordnet in same synsets. WordNet also provides short description and examples of the words. All synsets are connected to other synsets by means of semantic relations like hypernyms, hyponyms etc. The hypernym/hyponym relationships among the noun synsets can be interpreted as specialization relations among conceptual categories.

Relation	Description	Example
Hypernym	is a generalization of	motor vehicle is a hypernym of car
Hyponym	is a kind of	car is a hyponym of motor vehicle
Meronym	is a part of	lock is a meronym of door
Holonym	contains part	door is a holonym of lock
Troponym	is a way to	fly is a troponym of travel
Antonym	opposite of	stay in place is an antonym of travel
Attribute	attribute of	fast is an attribute of speed
Entailment	entails	calling on the phone entails dialing
Cause	cause to	to hurt causes to suffer

Table 2.1: Relations in WordNet.

CHAPTER 3
SEMANTIC MATCHING

The match operator takes two graph-like structures and produces a mapping between the nodes of the graphs that correspond semantically to each other.

With the emergence of the semantic web and the growing number of heterogenous data sources, the benefits of ontologies are becoming widely accepted. The domain of application is widening every day, ranging from word sense disambiguation to search of biological macromolecules such as DNA and proteins.

Many diverse solutions of match have been proposed so far. We focus on a schema-based solution, namely a matching system exploiting only the schema information, thus not considering instances. We follow a novel approach called semantic matching [1]. Semantic matching is a technique used in computer science to identify information which is semantically related. This approach is based on the two key ideas. The first is that we calculate mappings between schema elements by computing semantic relations (e.g., equivalence, more generality, disjointness), instead of computing coefficients rating match quality in the [0,1] range, as it is the case in the most previous approaches, see, for example, [7–9]. The second idea is that we determine semantic relations by analyzing the meaning (concepts, not labels) which is codified in the elements and the structures of schemas. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the labels intended meaning. This allows us to translate the matching problem into a propositional unsatisfiability problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability (SAT) deciders, e.g., [10].

3.1 Related Study

At present, there exists a line of semi-automated schema matching systems, see, for instance [7–9, 11] . A good survey and a classification of matching approaches up to 2001 is provided in [3], while an extension of its schema-based part and a user-centric classification of matching systems is provided in [13]. In particular, for individual matchers, [13] introduces the following criteria which allow for detailing further (with respect to [3]), the element and structure level of matching: syntactic techniques (these interpret their input as

a function of their sole structures following some clearly stated algorithms, e.g., iterative fix point computation for matching graphs), external techniques (these exploit external resources of a domain and common knowledge, e.g., WordNet [14]), and semantic techniques (these use formal semantics, e.g., model-theoretic semantics, in order to interpret the input and justify their results). The distinction between the hybrid and composite matching algorithms of [3] is useful from an architectural perspective. [13] extends this work by taking into account how the systems can be distinguished in the matter of considering the mappings and the matching task, thus representing the end-user perspective. In this respect, the following criteria are proposed: mappings as solutions (these systems consider the matching problem as an optimization problem and the mapping is a solution to it, e.g., [9]); mappings as theorems (these systems rely on semantics and require the mapping to satisfy it, e.g., the approach proposed in this paper); mappings as likeness clues (these systems produce only reasonable indications to a user for selecting the mappings, e.g., [7, 8]). Let us consider some recent schema-based state of the art systems in light of the above criteria.

RONDO: The Similarity Flooding (SF) [9] approach, as implemented in Rondo, utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs. The algorithm exploits only syntactic techniques at the element and structure level. It starts from the string-based comparison (common prefixes, suffixes tests) of the nodes labels to obtain an initial mapping which is further refined within the fix-point computation. SF considers the mappings as a solution to a clearly stated optimization problem.

CUPID: Cupid [8] implements a hybrid matching algorithm comprising syntactic techniques at the element (e.g., common prefixes, suffixes tests) and structure level (e.g., tree matching weighted by leaves). It also exploits external resources, in particular, a precompiled thesaurus. Cupid falls into the mappings as likeness clues category.

COMA: COMA [7] is a composite schema matching system which exploits syntactic and external techniques. It provides a library of matching algorithms; a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. The matching library is extensible, it contains 6 elementary matchers, 5 hybrid matchers, and one reuse-oriented matcher. Most of them implement string-based techniques (affix, n-gram, edit distance, etc.); others share techniques with Cupid (tree matching weighted by leaves, thesauri look-up, etc.); reuse-oriented is a completely novel matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Distinct features of COMA with respect to Cupid, are a more flexible architecture and a possibility of performing iterations in the matching process. COMA falls into the mappings as likeness clues category.

Figure 3.1: Parts of Google and Yahoo directories integrate these two directories

3.2 Motivating Scenario

In order to motivate the matching problem and illustrate one of the possible situations which can arise in the data integration task let us use the (parts of the Google and Yahoo) directories depicted in Figure 3.1.

A first step in the integration process is to identify the matching candidates. For example, ShoppingO1 can be assumed equivalent to ShoppingO2, while Board GamesO1 is less general than GamesO2. Hereafter the subscripts designate the directory (either O1 or O2) of the node considered. Once the correspondences between two schemas have been determined, the next step has to generate query expressions that automatically translate data instances of these schemas under an integrated schema. We think of a mapping element (or mapping) as a 4-tuple ($ID_{i,j}$, $n1_i$, $n2_j$, R_i).

$$i = 1, 2, 3 \ldots N1$$

$$j = 1, 2, 3 \ldots N2$$

where $ID_{i,j}$ is a unique identifier of the given mapping element; $n1_i$ is the i-th node of the first graph, N1 is the number of nodes in the first graph; $n2_j$ is the j-th node of the second graph, N2 is the number of nodes in the second graph; and R specifies a similarity relation of the given nodes. For instance, in this paper we consider equivalence ($\equiv$); more general ($\supseteq$); less general ($\subseteq$); disjointness ($\perp$) relations. The semantics of the above relations are the obvious set-theoretic semantics. When none of the relation holds, the special Idk (I do not know) relation is returned. We define matching as the process of discovering mappings between two graph-like structures through the application of a matching algorithm.

3.3 Element Level Semantic Matching

The matching process is often articulated into two basic steps, namely element and structure level matching [3].

Element level matchers consider only the information at the atomic level (e. g., the information contained in elements of the schemas);

Structure level matchers often aggregate the results of the several element level matchers and consider also the information about the structural properties of the schemas.

The matching process is often articulated into two basic steps, namely element and structure level matching. Element level semantic matchers return semantic relations ($\subseteq$, $\supseteq$, $\perp$, $\equiv$, Idk) rather then similarity coefficients [0..1] which are often considered as equivalence relation with certain level of plausibility or confidence [1].

3.3.1 Knowledge Based Matchers

Knowledge based matchers take in input two concept (or synset) identifiers defined in WordNet. They produce semantic relations by exploiting its structural properties. In some cases they combine the knowledge derived from WordNet with statistics collected from large scale corpora.

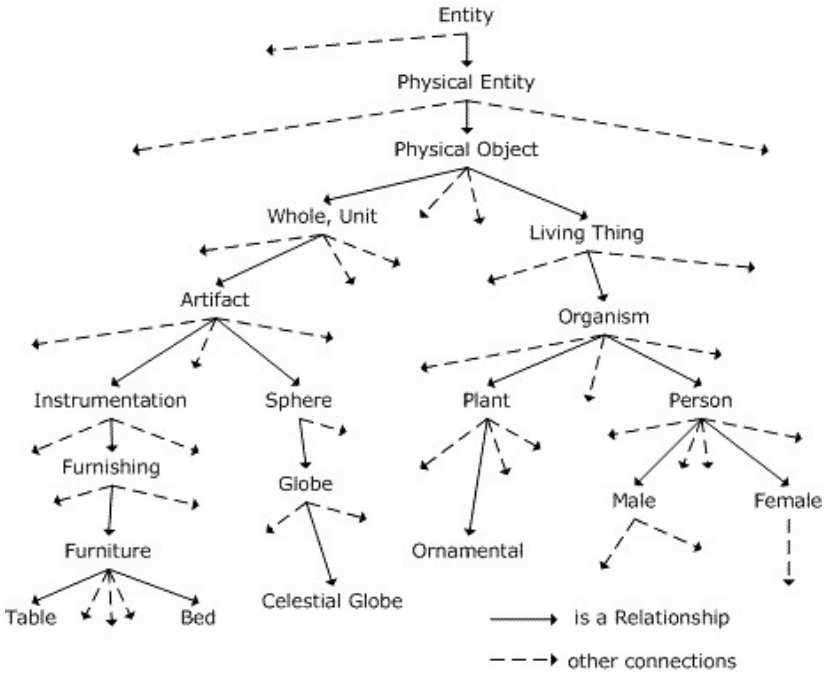

Figure 3.2: An example of WordNet nouns taxonomy

Often knowledge based matchers are based on either similarity or relatedness measures. If the value of the measure exceeds the given threshold the certain semantic relation is produced. Otherwise Idk is returned.

The wordnet matcher is a well known knowledge based matcher. It translates the relations provided by WordNet to semantic relations according to the following rules: A $\subseteq$ B if A is a hyponym, meronym or troponym of B; A $\supseteq$ B if A is a hypernym or holonym of B; A $\equiv$ B if they are connected by synonymy relation or they belong to one synset (night and nighttime from abovementioned example); A $\perp$ B if they are connected by antonymy relation or they are the siblings in the part of hierarchy.

Notice that hyponymy, meronymy, troponymy, hypernymy and holonymy relations are transitive. Therefore, for example, from Figure 3.2 we can derive that Person $\subseteq$ LivingThing. If none of the abovementioned relations holds among the two input synsets Idk relation is returned.

Source lable	Target lable	Semantic Relation
car	minivan	$\supseteq$
car	auto	$\equiv$
tail	dog	$\subseteq$
red	pink	idk

Table 3.1: An Example of WordNet nouns taxonomy.

Table 3.3.1 illustrates WordNet matcher results.

CHAPTER 4
STRING METRIC BASED SIMILARITY

String metric based similarity is done comparing two string depending on their distance. A string metric provides a number indicating an algorithm-specific indication of distance. String metrics are used heavily in information integration and are currently used in areas including fraud detection, fingerprint analysis, plagiarism detection, ontology merging, DNA analysis, RNA analysis, image analysis, evidence-based machine learning, database data deduplication, data mining, Web interfaces, e.g. Ajax-style suggestions as you type, data integration, and semantic knowledge integration [15].

4.1 Related Study

Dozens of ontology matching systems have been developed over the last decade, and nearly all of them use of a string similarity metric. But despite their ubiquity, there has been little systematic analysis on which metrics perform well when applied to ontology matching. Today quite a lot of string metrics exist in literature. These string metrics have been developed and applied in different scientific fields like statistics, for probabilistic record linkage [16], database, for record matching [17], Artificial Intelligence, for supervised learning [18] and Biology, for identifying common molecular subsequences [19].

Recent work by Ngo and his colleagues has analyzed the performance of some string metrics for ontology matching and their interaction with structural and semantic metrics [20]. There has also been some prior analysis of string similarity metrics in the context of ontology matching as part of the development of a new string similarity metric designed specifically for this domain done by Stoilos and his colleagues [21]. They compared the performance of a variety of string metrics on a subset of the Ontology Alignment Evaluation Initiative (OAEI) benchmark test set and determined that the performance among metrics varied considerably. Another piece of work done in this area is a report produced by the Knowledge Web Consortium in 2004 that contained a description of a variety string (terminological) metrics and string normalization and stemming applied to the problem of ontology alignment [22]. When the area of interest is expanded to include string similarity metric studies for other domains, we find some more interesting surveys. For instance, Branting looked at string similarity metrics as applied to name matching in legal case files [23]. He evaluated

the performance of various combinations of normalization, indexing (determining which names would be compared to one another) and similarity metrics. In addition, Cohen et al. did a very thorough analysis of string similarity metrics as applied to name matching tasks [24].

We can group string metrics along three major axes [25] : global versus local, set versus whole string, and perfect-sequence versus imperfect-sequence. Global versus local refers to the amount of information the metric needs in order to classify a pair of strings as a match or a non-match. Global metrics must compute some information over all of the strings in one or both ontologies before it can match any strings whereas for local metrics the pair of strings currently being considered is all the input that is required. Global metrics can be more tuned to the particular ontology pair being matched, but that comes at the price of increased time complexity. Perfect-sequence metrics require characters to occur in the same position in both strings in order to be considered a match. Imperfect sequence metrics equate matching characters as long as their positions in the strings differ by less than some threshold. In some metrics, this threshold is the entire length of the string. Imperfect-sequence metrics are thought to perform better when the word ordering of labels might differ but may result in more false positives. A set-based string metric works by finding the degree of overlap between the words contained in two strings. The set-based metric must still use a basic string metric to establish if the individual tokens are equal. Word-based set metrics are generally thought to perform well on long strings.

The list below contains all metrics found in the review of OAEI participants and categorizes them based on the classifications described above. For set-based metrics, the underlying base metric used is given in parentheses. A subset of these metrics (shown in bold) has been chosen for analysis related to various aspects of the ontology alignment problem. These metrics were chosen to reflect those most commonly used in existing alignment systems as well as to cover as fully as possible all combinations of the classification system provided.

4.1.1 Set, Global, Perfect-sequence

Evidence Content (with exact): Similar to Jaccard mentioned below, but words are weighted based on their evidence content (a function of their frequency in the ontology)

TF-IDF (with exact match): Term Frequency / Inverse Document Frequency; the idea is that two entities are more similar if they share a word that is rare in the ontologies

4.1.2 Set, Global, Imperfect-sequence

Soft TF-IDF (with Jaro Winkler)- A variant of TF-IDF that considers words equal based on Jaro Winkler (mentioned below) rather than exact match

4.1.3 Set, Local, Perfect-sequence

Jaccard (with exact match) - The number of words two strings have in common divided by the total number of unique words

Overlap Coecient (with exact) - The number of words two strings have in common divided by the number of words in the smaller string

4.1.4 Set, Local, Imperfect-sequence

RWSA - Redundant, Word-by-word, Symmetrical, Approximate; strings are indexed by their Soundex representation and are a match if each word in the smaller string has a weighted edit distance less than a threshold for a word in the longer string [23]

Soft Jaccard (with Levenstein) - Levenstein is run on all pairs of words in both strings and the number less than the threshold is counted and divided by the number of words in the longer string

4.1.5 Non-set, Global, Perfect-sequence

None

4.1.6 Non-set, Global, Imperfect-sequence

COCLU Compression-based Clustering; a Human tree is used to cluster the strings based on a metric called the Cluster Code Dierence, and strings in the same cluster are considered equivalent [26]

4.1.7 Non–set, Local, Perfect-sequence

Exact Match - Checks for string equality

Longest Common Substring - The length of the largest substring common to both strings, normalized by the length of the strings

Prex - Checks if the rst string is a prex of the second

Substring Inclusion - Whether the rst string is contained in the second

Sux - Whether the rst string is a sux of the second

4.1.8 Non-set, Local, Imperfect-sequence

Jaro - Based on the number of matching and transposed characters, where characters match if they are within a window based on the lengths of the strings and are transposed if they match but are in reverse order

Jaro Winkler - Variation of the Jaro metric that gives a preference to strings that share a common prex

Levenstein - The number of insertions, deletions, and substitutions required to transform one string to another; also called edit distance

Lin - Based on the idea that similarity between two strings can be determined by taking a measure of what they have in common and dividing by a measure of the information it takes to describe them [27]

Monge Elkan - A variant of Smith Waterman (see below) with nonlinear gap penalties, approximate character matching, and particular parameter values [17]

N-gram - Converts strings into sets of n-grams (we use n=3); the resulting sets are compared using a set similarity metric such as cosine similarity or Dices coecient

Normalized Hamming Distance - The number of substitutions required to transform one string into another, divided by the length of the string

Smith Waterman - Uses dynamic programming over a matrix describing the matches, insertions, and deletions between two strings

Smith Waterman Gotoh - A variant of the Smith-Waterman metric that has ane gap penalties

Stoilos - Specically developed for ontology alignment, this metric explicitly considers both the commonalities and the dierences of the strings being compared [21]

String Matching (SM) - A variant of Levenstein in which the dierence between the length of the shorter string and the edit distance is divided by the length of the shorter string [28]

Often alignment algorithms modify the strings before computing their similarity. All of the pre-processing approaches that were either tried or proposed by OAEI participants are listed here. The approaches mentioned by more than two participants are shown in bold - these will be examined in detail.

These approaches can be divided into two major categories: syntactic and semantic. Syntactic pre-processing methods are based on the characters in the strings or the rules of the

language in which the strings are written. They can generally be applied quickly and without reference to an outside data store. Semantic methods relate to the meanings of the strings. These methods generally involve using a dictionary, thesaurus, or translation service to retrieve more information about a word or phrase.

4.1.9 Syntactic

tokenization - Splitting strings into their component words based on delimiters and camelCase

split – Splitting the compound words

normalization - Elimination of stylistic dierences due to capitalization, punctuation, word order, and characters not in the Latin alphabet

stemming/lemmatization - Elimination of grammatical dierences due to verb tense, plurals, etc. We use the Porter stemming algorithm

stop word removal Removal of very common words. The Glasgow stop word list is used in this work

consider part-of-speech - Functional words such as articles, conjunctions, and prepositions are weighted less (or removed completely)

4.1.10 Semantic

synonyms - Strings are supplemented with their synonyms

antonyms - Used with metrics considering dierences and commonalities

categorization - An external source containing a category hierarchy is used. Strings falling into the same category are considered more similar.

language tag - Leverage language tags contained in some ontologies to avoid comparing labels in dierent languages or as an aid for translation

translations - Strings are translated before they are compared. We have used Google Translate.

expand abbreviations and acronyms - There have been several attempts to do such expansions into long form, by either looking them up in external knowledge sources or using language production rules.

4.2 Desired Properties

Ontology matching is a relatively new field in computer science. Thus, none of the classical string metrics has been created having the properties and characteristics of this field in mind. Algorithms that are used in ontology matching are very complex and contain many features and parameters that can affect the performance even of commonly accepted and good string metrics, when they are used in this new context. Features like the threshold (the value above which two pairs are considered identical), or the cardinality of mappings (one-to-one, one-to-many etc.) play a key role in ontology alignment and as we will see the metrics found in literature sometimes fail to give satisfactory results cause of the existence of these parameters. Thus, before we define our string metric we think that it is crucial to state the specifications that we want such a string metric to fulfill. More precisely we want the new metric to be:

1. **Fast**: Since ontologies are used in applications that demand processing in real-time, like the semantic web or intelligent retrieval tasks, the complexity of the string metric should be very low, leading to a fast matching procedure.

2. **Stable**: As we aforementioned, one very crucial parameter of ontology alignment algorithms is their threshold. When we will demand from alignment platform to automatically operate on the semantic web their threshold would probably be fixed at a value considered optimal by their authors. Though some methods that automatically adjust the threshold during runtime exist in literature [29] it cannot be proved that they select the optimum value for threshold each time an alignment is performed. Thus, we demand by the string metrics to be as stable as possible. By stability we define the ability of a string metric to perform almost optimal even if small diverges from the optimal threshold take place. As we will see all metrics fail to satisfy this crucial property. Even worst, classical metrics are really sensitive in small changes of the threshold, and while they can provide good results if the threshold is optimized, this performance can rapidly decrease if we slightly disturb the value of the threshold.

3. **Intelligent**: When operating for example in the semantic web context, it is likely that an ontology be compared to an irrelevant one, but with which string resemblances occur. In this case we want our metric to identify all the differences and provide us with correct results. But it is not uncommon the situation where usual string metrics fail to identify cases where two strings represent completely different concepts but resemble a lot. Consider for example the words score and store. They represent two completely different concepts. Though this is true the Monge-Elkan, Levenstein, SubString, Needleman-Wunsch, Q-Gram and Jaro-Winkler metrics rate the pair with a similarity degree of 0.68, 0.8, 0.6, 0.9, 0.57 and 0.88 which are relatively hight values.

4. **Discriminating**: One of the most usual cardinalities requested for alignment mappings is the one-to-one cardinality. As it is obvious in an one-to-one mapping if a string in a reference ontology is mapped with the same similarity degree to more than one in the second ontology it is very probable that the algorithm fails to pick the correct pair from the set of pairs. Hence, we would like from our similarity metric to rarely assign the same similarity degree when we compare one particular string to several others.

4.3 Jaro Winkler Distance

To improve the efficiency of the string matching results,a distance value between the given strings is added to the result. This distance is known as Jaro Winkler distance. The Jaro-Winkler distance is a measure of similarity between two strings. It is mainly used in the area of record linkage for duplicate detection. The higher the Jaro-Winkler distance for two strings, the more similar the strings are. The Jaro-Winkler distance metric is best suited for short strings such as person or proper names. The score is normalized such that 0 equates to no similarity and 1 is an exact match.

Given strings $s = a1 \ldots aK$ and $t = b1 \ldots bL$, define a character a_i in s to be $common$ with t there is a b_j = a_i in t such that $i - H \leq j \leq i + H$,where H=$min|s|, |t|/2$.

Let $s' = a'1 \ldots a'K$ be the characters in s which are common with t (in the same order they appear in s) and let $t' = b'1 \ldots b'L$, be analogous; now define a $transposition\ for\ s', t'$ to be a position i) such that $a'_i \neq b'_j$. Let $T_{s',t'}$ be half the number of transpositions for s' and t'. The Jaro similarity metric for s and t is

$$Jaro(s,t) = 1/3 \cdot (|S'|/|S| + |t'|/|t| + |s' - K|/|s'|$$

where K=$T_{s',t'}$

A variant of this due to Winkler [16] also uses the length P of the longest common prefix of s and t.

$$Jaro - Winkler(s,t) = Jaro(s,t) + |P|/|10| . (1 - Jaro(s,t))$$

CHAPTER 5
METHODOLOGY

5.1 Semantic Similarity Measurement

We use WordNet for semantic similarity measurement by employing WordNet based equality. WordNet based equality is derived from the senses of the words in WordNet. It states that two terms are equal if they have at least one sense in common, which is synonym of the second one [6]. In this process of equality measurement, senses of the words are inferred from WordNet and then are compared for finding common synonym. If any of the senses are common between two words, they are considered equal. The equality measure is denoted by Sim_{WN}. If t1 and t2 are two terms in consider, then the value of Sim_{WN}(t1 , t2) is 1.0 if at least one sense of t1 matches that of t2. Otherwise, if t1 and t2 get no senses in common between them, value of Sim_{WN}(t1 , t2) is 0.0. The expression for equality measure is :

$$\text{Sim}_{\text{WN}}(\text{t1 , t2}) = \begin{cases} 1.0, & \text{if } cond(t1, t2) holds \\ 0.0, & \text{otherwise} \end{cases}$$

where, cond(t1 , t2)= $\exists x\{x \in \text{senses(t1)} \wedge \text{senses(t2)}\}$

5.2 String Metric Measurements

The string metric is based on the intuitions presented in [27] about the similarity between two entities. We believe that the similarity between two entities is related to their commonalities as well as to their differences. Thus, the similarity should be a function of both these features. This feature also appears, sometimes implicitly, in other measures as well. For example, in those measures that perform string editing, such operation can be considered as a form of difference counting, while non-editing can be considered as similarity counting. Thus, our metric is defined by the following equation:

$$Sim(s1, s2) = Comm(s1, s2) - Diff(s1, s2) + winkler(s1, s2) \ldots (1)$$

where $comm(s1, s2)$ stands for the commonality between s1 and s2, $diff(s1, s2)$ for the difference and $WinklerImpr(s1, s2)$ for the improvement of the result using the method introduced by Winkler in [16]. We now have to define the functions of commonality and

difference. The function of commonality is motivated by the substring string metric. In the substring metric the biggest common substring between two strings is computed. This process is further extended by removing the common substring and by searching again for the next biggest substring until no one can be identified. The sum of the lengths of these substrings is then scaled with the length of the strings. The intuition behind this extension of the substring metric is the following. In the field of Computer Science researchers tend to use descriptive names for their variables or the units that represent real world entities. In other cases they tend to concatenate words and create new ones. For example in order to represent the concept of the number of pages of a book it is likely that someone uses the word numberOfPages or some one else might use the word numPages. As one can see these two strings share not one but two common substrings which is very crucial to identify in order to approximate their real similarity as much as possible. Moreover, we can now distinguish cases like the above with cases where the substring Pages is shared but the rest of the strings are quite different, thus satisfying the specification for an intelligent metric. Hence, the function of commonality is given by the following equation:

$$Comm(s1, s2) = 2 * ilength(maxComSubStringi)length(s1) + length(s2) \ldots (2)$$

As for the difference function, this is based on the length of the unmatched strings that have resulted from the initial matching step. Moreover, we believe that difference should play a less important role on the computation of the overall similarity. Our choice was the Hamacher product [30], which is a parametric triangular norm. This leads us to the following equation:

$$Diff(s1, s2) = uLens1 * uLens2p + (1 - p) * (uLens1 + uLens2 - uLens1 * uLens2) \ldots (3)$$

where $p \in$ [0,8), and uLens1, uLens2 represent the length of the unmatched substring from the initial strings s1 and s2 scaled with the string length, respectively.

CHAPTER 6
EXPERIMENTS AND RESULTS

6.1 Semantic Similarity

By applying WordNet based equality for semantic similarity measures, we get the measurement result in nominal form. The result is either "equal" or "non-equal". "Equal" value is denoted by 1.0 and "non-equal" value is denoted by 0.0.

6.1.1 Code Analysis

For our thesis work we used Python programming language. We connected the "Wordnet Database" with our code by importing "NLTK" package of python. It enabled us to access the database according to our will. We start our program by taking input two desired words. The program takes the input and find the synsets of the words. Afterwards it finds the common synsets between the two of them and provides the common synset of the words. The code for Semantic matching:

```
1 from nltk.corpus import wordnet as wn
2 import re
3 print('Enter␣first␣Word')
4 word1=input()
5 print('Enter␣2nd␣Word')
6 word2=input()
7 str1=wn.synsets(word1)
8 str2=wn.synsets(word2)
9 print('The␣SYNSETS␣OF',word1)
10 print(str1)
11 print('The␣SYNSETS␣OF',word2)
12 print(str2)
13 print(len(str1))
14 print(len(str2))
15 x='nothing'
16 y='nothing'
```

```
for a in str1:
    if a in str2:
        x=a
    else:
        y=a
print('Matches are:',x)
```

6.1.2 Experiment-1

Let us consider two noun terms “forest” and “wood”. In WordNet the word “forest” gives us two senses:

1. **forest, wood, woods** - the trees and other plants in a large densely wooded area

2.**forest, woodland, timberland, timber** - land that is covered with trees and shrubs

The word “wood” gives us eight senses from WordNet:

1. **wood** - the hard fibrous lignified substance under the bark of trees

2. **forest**, **woodland**, **timberland**, timber - land that is covered with trees and shrubs

3. **Wood**, **Natalie Wood** - United States film actress (1938-1981)

4. **Wood, Sir Henry Wood, Sir Henry Joseph Wood** - English conductor (1869-1944)

5. **Wood, Mrs. Henry Wood, Ellen Price Wood** - English writer of novels about murders and thefts and forgeries (1814-1887)

6. **Wood, Grant Wood** - United States painter noted for works based on life in the Midwest (1892-1942)

7. **woodwind, woodwind instrument, wood** - any wind instrument other than the brass instruments

8. **wood** - a golf club with a long shaft used to hit long shots; originally made with a wooden head; metal woods are now available

Here wee see that, sense 1 of the term “forest” exactly matches with the sense 2 of the term “wood”. So the value of Sim_{WN}(t1 , t2) becomes 1.0 for this case since one sense is common between the terms in consider. So the terms “forest” and “wood” are to be said “semantically equal”.

6.1.3 Experiment-2

For the second case study, let us consider two adjective terms “angry” and “violent” in WordNet.

The word “angry” gives following three senses:

1. **angry** - feeling or showing anger

2. **angry, furious, raging, tempestuous, wild** - (of the elements) as if showing violent anger

3. **angry** - severely inflamed and painful

Though “violent” seems similar to “angry”, let us take a look at the senses it shows in WordNet:

1. **violent** - acting with or marked by or resulting from great force or energy or emotional intensity; “a violent attack”; “a violent person”; “violent feelings”

2. **violent** - effected by force or injury rather than natural causes; “a violent death”

3. **violent, wild** - (of colors or sounds) intensely vivid or loud; “a violent clash of colors”; “her dress was a violent red”; “a violent noise”; “wild colors”; “wild shouts”

4. **fierce, tearing, vehement, violent, trigger-happy** - marked by extreme intensity of emotions or convictions; inclined to react violently; fervid; “fierce loyalty”; “in a tearing rage”; “vehement dislike”; “violent passions”

5. **crimson, red, violent** - characterized by violence or bloodshed; “Writes of crimson deeds and barbaric days”- Andrea Parke; “ Fann’d by Conquest’s crimson wing”- Thomas Gray; “Convulsed with red rage”- Hudson Strode

Here we see that no senses of “angry” and “violent” matches with one another. In according to our applied method Sim_{WN}(t1 , t2) is equal to 0.0 which means these two terms are semantically unequal.

6.2 String Metric Similarity

For string metric similarity measurement, we are finding the length of common substring and uncommon substring between two words. We use Jaro Winkler distance with our results to obtain string similarity between the two given terms.

6.2.1 Code Analysis

For string matching we take input two words. We find the common substring of both the words. We take the input for the value of P. After that we find the difference of the total length of the word and common substring length to obtain the unmatched substring length. Using the values of common substring length and unmatched substring length we obtain the commonality and difference value. Then we call the function "jaro.distance" which is imported from the package jellyfish with both words as parameters. It provides the Jaro distance value. We then calculate the similarity using the formula.

The code we implemented is:

```
import jellyfish

print('Enter␣first␣Word')
word1=input()
print('Enter␣second␣Word')
word2=input()
print ('Give␣the␣value␣of␣P')
p1=input()
p=float(p1)
len1=len(word1)
len2=len(word2)
answer=""
for i in range(len1):
    for j in range(len2):
        if(i==j and word1[i]==word2[j]):
            answer=answer+word1[i]

commonlength=len(answer)
unmatch1=len1-commonlength
unmatch2=len2-commonlength
comm=(2*commonlength)/(len1+len2)
difference=(unmatch1*unmatch2)/(p+(p-1)*(unmatch1+unmatch2-(unmatch1*un
jarodistance=jellyfish.jaro_distance(word1,word2)
result=comm-difference+jarodistance
print(result)
```

6.2.2 Experiment 1

Let us consider two word “Attribute” as s1 and “Attraction” as s2, if we want to find similarity by string metric measurement between these two word. The equation for this measurement is Sim(s1, s2) = Comm(s1, s2) - Diff(s1, s2) + winkler(s1, s2).

At first find the letters which are common in both letter.Here 4 letters are common. They are “attr”.

Now we have to measure commonality between these words by applying bellow equation:

Comm(s1, s2) = 2 * i length(maxComSubStringi)length(s1) + length(s2) For these words the result is: 0.42105263157894735.

Then we measure the difference between these two words by applying this equation:

Diff(s1, s2) = uLens1 * uLens2 p + (1 - p) * (uLens1 + uLens2 - uLens1 * uLens2)

The result: 1.7441860465116277.

Here we consider the value of p is: 0.1.

For the improvement of the result we have to calculate the method introduced by Winkler.

The result of Jaro Winkler method: .70001.

The total result of similarity: -0.62312341493268035

6.2.3 Experiment 2

Let us consider two word “Price” as s1 and “Pride” as s2, if we want to find similarity by string metric measurement between these two word.

At first find the letters which are common in both letter. Here 4 letters are common. They are p,r,i,e.

Now we have to measure commonality between these words by applying bellow equation:

For these words the commonality result is: 0.8.

The result of difference is: -1.25.

Here we consider the value of p is: 0.1.

For the improvement of the result we have to calculate the method introduced by Winkler.

The result of Jaro Winkler method is: .86666.

The total result of similarity is: 2.91666

CHAPTER 7
DISCUSSION

By doing our experiments we found out the semantic matching of two words based on their synsets. We got the result on the basis matching senses. Our results shows the matchings between the two words. If there is at least one sense is common between the words the result shows “MATCHED”. If there is no match between the two words we get the result as “NO MATCH”.

We also experimented on String matching. It provides us results based on the commnality and difference. Generally the result is a positive number. But sometimes it is seen that if two words have less commnality than difference then the result shows negative value.

We found that the result of string matching was not too perfect. So to improve our result we also added Jaro Winkler Distance to the formula. Jaro Winkler Distance requires a value entitled as P which can be varied from 0 to 8. In our experiment we considered the value of P is 0.1.

Working on our thesis we achieved the semantic similarities of two words basing on their senses. It is seen that two words can be assumed to be same in meaning but if they are considered by senses,they are not actually same. Our semantic matching distinguishes between such kind of words. String matching is quite similar like matching substrings between two words. But by simply finds the common and uncommon substring we can not get the perfect similarity. So we have to implement the Jaro Distance as a improvement for the result.

7.1 Future Expansion

In future we plan to implement semantic matching and string matching in Bangla language. It would be a great challenge as there is no WordNet Database on Bangla and Bangla language has far different grammatical structure than English.

7.2 Conclusion

Our thesis experiment was based on similarity features between two words. To achieve it we implemented semantic and string matching between the words. Semantic matching occupies a vast sector of matching algorithms, we just gained partial knowledge on this topic. String matching is another similarity that we have implemented for our thesis. It also uses Jaro Winkler Distance for improvement of the result. We achieved the value for both the similarities between two words in our thesis. This enabled us to find how two words have different similarities based on String Matching and Semantic Matching.

REFERENCES

[1] F. Giunchiglia and P. Shvaiko, "Semantic matching," *Knowledge Eng. Review*, vol. 18, no. 3, pp. 265–280, 2003.

[2] http://wordnet.princeton.edu/.

[3] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *the VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001.

[4] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *Proeceedings of the 11th International Conference on World Wide Web*, pp. 662–673, 2002.

[5] N. F. Noy and M. A. Musen, "Anchor-prompt: Using non-local context for semantic matching," in *Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial intelligence (IJCAI)*, pp. 63–70, 2001.

[6] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, *S-Match: an algorithm and an implementation of semantic matching*. Springer, 2004.

[7] H.-H. Do and E. Rahm, "Coma: a system for flexible combination of schema matching approaches," in *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 610–621, VLDB Endowment, 2002.

[8] P. A. Bernstein, J. Madhavan, and E. Rahm, "Generic schema matching, ten years later," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 695–701, 2011.

[9] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and ist application to schema matching," in *Proc. 18th ICDE*, (San Jose, CA), Feb. 2002.

[10] http://www.sat4j.org/.

[11] S. Bergamaschi, S. Castano, and M. Vincini, "Semantic integration of semistructured and structured data sources," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 28, no. 1, pp. 54–??, 1999. Special section on semantic interoperability in global information systems.

[12] E. Rahm and P. Bernstein, "A survey of approaches to automatic schema matching," 2001.

[13] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," vol. 3730, pp. 146–171, 2005.

[14] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, pp. 39–41, Nov. 1995.

[15] http://en.wikipedia.org/wiki/String_metric/.

[16] W. E. Winkler, "The state of record linkage and current research problems," in *Statistical Research Division, US Census Bureau*, Citeseer, 1999.

[17] A. E. Monge, C. Elkan, *et al.*, "The field matching problem: Algorithms and applications.," in *KDD*, pp. 267–270, 1996.

[18] S. Tejada, C. A. Knoblock, and S. Minton, "Learning object identification rules for information integration," *Information Systems*, vol. 26, no. 8, pp. 607–633, 2001.

[19] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[20] D. Ngo, Z. Bellahsene, and K. Todorov, "Opening the black box of ontology matching," in *The Semantic Web: Semantics and Big Data*, pp. 16–30, Springer, 2013.

[21] G. Stoilos, G. Stamou, and S. Kollias, "A string metric for ontology alignment," in *The Semantic Web–ISWC 2005*, pp. 624–637, Springer, 2005.

[22] J. Euzenat, T. Le Bach, J. Barrasa, P. Bouquet, J. De Bo, R. Dieng-Kuntz, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, *et al.*, "State of the art on ontology alignment," *Knowledge Web Deliverable D*, vol. 2, pp. 2–3, 2004.

[23] L. Branting, "A comparative evaluation of name-matching algorithms," p. 224232, ACM, 2003.

[24] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *KDD Workshop on Data Cleaning and Object Consolidation*, vol. 3, pp. 73–78, 2003.

[25] M. Cheatham and P. Hitzler, "String similarity metrics for ontology alignment," in *The Semantic Web–ISWC 2013*, pp. 294–309, Springer, 2013.

[26] A. G. Valarakos, G. Paliouras, V. Karkaletsis, and G. Vouros, "A name-matching algorithm for supporting ontology enrichment," in *Methods and Applications of Artificial Intelligence*, pp. 381–389, Springer, 2004.

[27] Lin and Dekang, "An information-theoretic definition of similarity," in *ICML*, vol. 98, pp. 296–304, 1998.

[28] A. Maedche and S. Staab, “Measuring similarity between ontologies,” in *Knowledge engineering and knowledge management: Ontologies and the semantic web*, pp. 251–263, Springer, 2002.

[29] M. Ehrig and Y. Sure, “Ontology mapping–an integrated approach,” in *The Semantic Web: Research and Applications*, pp. 76–91, Springer, 2004.

[30] H. Hamacher, H. Leberling, and H.-J. Zimmermann, “Sensitivity analysis in fuzzy linear programming,” *Fuzzy sets and systems*, vol. 1, no. 4, pp. 269–281, 1978.

APPENDIX A
CODES

A.1 Codes

Here are the codes that are implemented in our work.

A.1.1 Semantic Matching

We use this code to find out the Semantic Matching...

```
from nltk.corpus import wordnet as wn
import re
print('Enter first Word')
word1=input()
print('Enter 2nd Word')
word2=input()
str1=wn.synsets(word1)
str2=wn.synsets(word2)
print('The SYNSETS OF',word1)
print(str1)
print('The SYNSETS OF',word2)
print(str2)
print(len(str1))
print(len(str2))
x='nothing'
y='nothing'
for a in str1:
    if a in str2:
        x=a
    else:
        y=a
print('Matches are:',x)
```

A.1.2 String Matching

We use this code to find out the String Matching...

```
import jellyfish

print('Enter␣first␣Word')
word1=input()
print('Enter␣second␣Word')
word2=input()
print ('Give␣the␣value␣of␣P')
p1=input()
p=float(p1)
len1=len(word1)
len2=len(word2)
answer=""
for i in range(len1):
    for j in range(len2):
        if(i==j and word1[i]==word2[j]):
            answer=answer+word1[i]

commonlength=len(answer)
unmatch1=len1-commonlength
unmatch2=len2-commonlength
comm=(2*commonlength)/(len1+len2)
difference=(unmatch1*unmatch2)/(p+(p-1)*(unmatch1+unmatch2-(unmatch1*un
jarodistance=jellyfish.jaro_distance(word1,word2)
result=comm-difference+jarodistance
print(result)
```