Modern Applied Science; Vol. 6, No. 10; 2012
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

A Terminological Search Algorithm for Ontology Matching

Ahmad Zaeri[1] & Mohammad Ali Nematbakhsh[1]

[1] Computer Department, University of Isfahan, Isfahan, Iran

Correspondence: Ahmad Zaeri, Computer Department, University of Isfahan, Hezar Jarib Street, Isfahan, Iran. Tel: 98-311-793-4010. E-mail: zaeri@eng.ui.ac.ir

Received: August 6, 2012 Accepted: September 16, 2012 Online Published: September 26, 2012

doi:10.5539/mas.v6n10p37 URL: http://dx.doi.org/10.5539/mas.v6n10p37

Abstract

Most of the ontology alignment tools use terminological techniques as the initial step and then apply the structural techniques to refine the results. Since each terminological similarity measure considers some features of similarity, ontology alignment systems require exploiting different measures. While a great deal of effort has been devoted to developing various terminological similarity measures and also developing various ontology alignment systems, little attention has been paid to develop similarity search algorithms which exploit different similarity measures in order to gain benefits and avoid limitations. We propose a novel terminological search algorithm which tries to find an entity similar to an input search string in a given ontology. This algorithm extends the search string by creating a matrix from its synonym and hypernyms. The algorithm employs and combines different kind of similarity measures in different situations to achieve a higher performance, accuracy, and stability in comparison with previous methods which either use one measure or combine more measures in a naive ways such as averaging. We evaluated the algorithm using a subset of OAEI Bench mark data set. Results showed the superiority of proposed algorithm and effectiveness of different applied techniques such as word sense disambiguation and semantic filtering mechanism.

Keywords: terminological search, similarity measures, semantic similarity, ontologies matching

1. Introduction

The vision of Semantic Web is about having data and knowledge machine understandable so that machines can analyze and process complex request of humans more efficiently. In other words, the semantic web should facilitate information sharing in any form and integrate information from different sources such as web contents or database records (Berners-Lee, Hendler, & Lassila, 2001). An initial step toward this vision has been taken by representing terminologies of different domains as Ontologies. Even in the same domain, having different ontologies cannot be avoided due to complexity or expansiveness of knowledge or contrasting and distinctive views of different users (Romero, Vázquez-Naya, Loureiro, & Ezquerra, 2009). Consequently, to successfully integrating data sources with different ontologies, it is needed to align their ontologies through a process called Ontology Alignment. In its simplest form, an ontology alignment is finding of one to one correspondences among entities of two ontologies. In recent years, a large number of ontology alignment systems have been developed to detect such correspondences.

These systems usually employ different alignment techniques which have been categorized by Euzenat et al. (2007) in four main classes: terminological, structural, extensional, and semantic techniques. Terminological techniques try to find correspondences by investigating similarities between entities names. Alternatively, the structural methods consider internal structure of an entity or its relations to other entities as a source of detecting correspondences. Most of the developed alignment tools use terminological techniques as the initial and the main alignment approach and then apply the structural techniques to refine the results and to improve the accuracy. Besides these two main techniques, some systems use extensional and semantic techniques. Extensional techniques are inspired by the idea that having more commonalities between two entities' individuals (i.e., instances) might imply higher probability of matching between them. Although semantic techniques usually employ theoretical models and deduction to find similarity between interpretations of entities, the inductive nature of ontology alignment makes using of such deductive techniques difficult. Thus, semantic techniques are mostly utilized for validation of detected correspondences (Euzenat & Shvaiko, 2007).

Terminological techniques are divided into two main groups, the *string based* and the *language based*

approaches. The first one are techniques that just consider entity names as a sequence of characters and assume that higher similarity between structures of two sequences shows a higher similarity between them. For example, these techniques consider *PublishedBy* highly similar to *Publisher*, whereas they distinguish *Paper* from *Article*. The second one considers occurrence of some meaningful words in the names and compare two names by similarity between the meaning of those words (e.g., by using some external thesaurus). For example, these approaches can easily detect similarity between *Paper* and *Article*.

However, each of these measures has its own cons and pros. For example, string based measures usually have higher computational speeds and are more immune to small morphological changes (e.g., Food and Foods), but they are more sensitive to none morphological changes (e.g., *Food* and *Flood*). On the other hand, language based measures are less sensitive to none morphological changes (e.g., difference between *Food* and *Flood* can be easily detected), but they are much slower and are more sensitive to morphological changes. Although stemmer algorithms could help to find the root of a word, it would not be so beneficial if morphological changes highly alter the meaning of the word (e.g., *rewriter* and *writing*).

Each terminological similarity measure considers some aspects or features of similarity, so ontology alignment systems require exploiting different measures to achieve higher accuracy. Based on using similarity measures, ontology alignment systems can be divided into two groups: First, alignment systems which directly use similarity measures or combine different measures by some special technique such as averaging or weighted averaging; Second, alignment systems which independently developed their own techniques without using well known terminological measures.

While a great deal of effort has been devoted to developing various terminological similarity measures and also developing various ontology alignment systems, little attention has been paid to develop similarity search algorithms which exploit different similarity measures in order to gain their benefits and avoid their limitations.

In this paper, we suggest a novel terminological similarity search algorithm to find a concept Name (property or individual) in an ontology that is similar to a search string *SimString*. This algorithm extracts all synonyms and hypernyms of *SimString* from the WordNet and then creates a matrix which each row of this matrix represents one meaning of *SimString*. In fact, this algorithm not only tries to find concepts similar to *SimString* but also tries to find concepts similar to synonyms and hypernyms of all different *SimString* meanings as well. In addition, this algorithm prioritizes the synonyms and hypernyms in which it considers more specific words firstly then tries more general ones, so it can handle the situations that besides equal concepts, super concepts are interested. Once all ontology concepts names were compared to matrix candidates, each row of matrix, which represents one meaning of *SimString*, is scanned to find the at most one candidate per each row. This algorithm applies special kind of semantic filtering by removing all selected candidates that have a semantic similarity to *SimString* less than a threshold. The JIANG similarity measure has been used in the filtering part. In other words, we suppose that having a high semantic similarity is necessary but not sufficient to consider two words similar. ISub (Stoilos, Stamou, & Kollias, 2005) lexical similarity measure, as reported, has a clear superiority in ontology name searching over other measures such as Levenstein. Hence, ISub has been used in calculating of syntactical similarity calculation. Finally, algorithm uses a word sense disambiguation based on averaging to select the final similar concept (property or individual). This algorithm employs and combines different kind of similarity measures in different situations to achieve a higher performance, accuracy, and stability in comparison with either using one measure separately or combining different measures in simple ways such as averaging.

The rest of this article is structured as follows. After giving an overview of related work in Section 2, we illustrate the main algorithm in Section 3. Following this, Section 4 provides details on the experiments that we carried out and the results achieved. In these experiments we showed the superiority of our algorithm by using bench mark data set (Euzenat, Meilicke, Stuckenschmidt, Shvaiko, & dos Santos, 2011). We conclude with discussion of the approach and topics for future work in Section 5.

2. Related Work

Today, large numbers of ontology alignment systems exist in literature. Most of these alignment systems use terminological techniques as the first step to find the correspondences between ontologies' entities. These techniques usually utilize similarity measures to find such correspondences. Different similarity measures have been proposed to exploit different aspects of similarities between entities. As mentioned earlier, in literature, terminological similarity measures are usually divided into *string based* and *language based* measures.

Most widely used string based measure is the *Levenshtein distance* (Levenshtein, 1966). The Levenshtein distance between two strings is the minimum number of insertion, deletion, or substitution of a single character needed to transform one string into the other. Commonly used in bioinformatics, *Needleman-Wunsch distance*

(Needleman & Wunsch, 1970) is the modified version of Levenshtein distance which applies higher weight to insert and delete operations in comparison with substitution. *Jaro distance* (1995) and its modified variant *Jaro-Winkler distance* (1999) have been proposed for matching the names that may contain some spelling errors. Jaro measure considers common character between two strings and their order to calculate the similarity. Jaro measure is defined as follow (Euzenat & Shvaiko, 2007):

$$\text{Jaro}(s_1, s_2) = \frac{1}{3} \times \left(\frac{|\text{com}(s_1, s_2)|}{|s_1|} + \frac{|\text{com}(s_2, s_1)|}{|s_2|} + \frac{|\text{com}(s_1, s_2)| - |\text{transp}(s_1, s_2)|}{|\text{com}(s_1, s_2)|} \right) \quad (1)$$

which $com(s_1,s_2)$ is the common character between s_1 and s_2 and $transp(s_1,s_2)$ is the characters in $com(s_1, s_2)$ with different orders in s_1 and s_2.

Another feature for calculating similarity is to consider the number of common substrings between two strings.

$$subsim(s_1, s_2) = \frac{2 \times |maxCommonSubstring(s_1, s_2)|}{|s_1| + |s_2|} \quad (2)$$

Stoilos et al. (2005) have proposed ISUB measure which has specially been designed for ontology alignment by extending the idea of subsim measure:

$$ISUB(s_1, s_2) = Comm(s_1, s_2) - Diff(s_1, s_2) + winkler(s_1, s_2) \quad (3)$$

which

$$Comm(s_1, s_2) = \frac{2 \times \sum_i |maxComSubString_i|}{|s_1| + |s_2|} \quad (4)$$

and $maxCommonSubstring_i$ extends the idea of maximum common substring by considering the next common substrings after removing previous common substrings.

$$Diff(s_1, s_2) = \frac{uLen(s_1) * uLen(s_2)}{0.6 + 0.4 \times (uLen(s_1) + uLen(s_2) - uLen(s_1) * uLen(s_2))} \quad (5)$$

$uLen$ represents the length of the unmatched substring from the initial strings. $winkler(s_1, s_2)$ is the Jaro-Winkler similarity measure added for extra improvement.

They argue that for the case of ontology matching ISUB has a higher performance in comparison with other string based measures in the term of F1, precision, and recall (Stoilos et al., 2005).

In contrast to *string based measures*, *language based measures* consider string in the word level other than character level. In literature, language based measures are included two main groups: *intrinsic* and *extrinsic*. Intrinsic measures employ some linguistics techniques such as stemming, removing stop words, and part of speech tagging to find similarities between words while extrinsic measures uses some external resources such as dictionary and thesaurus to match words by considering their meaning. Many of extrinsic measures in ontology world utilize WordNet (Fellbaum, 1998) as an external resource to calculate extrinsic similarity measures (Euzenat & Shvaiko, 2007).

The WordNet extrinsic measures are divided into the three categories based on the kind of information that they consider: *path based*, *information content based*, and *hybrid measures*.

Path based measures: These measures use distance between two words' node in the taxonomy graph and their place to calculate the similarity. Higher distance between two nodes shows lower similarity between words.

Rada et al. (Rada, Mili, Bicknell, & Blettner, 1989) uses the length of path between two concepts as the distance between two concepts.

$$radaDistance = length(path(C_1, C_2)) \quad (6)$$

Leacock and Chodorow (1998) have normalized the Rada distance by using a D factor as the depth of the taxonomy contains the two concepts. Then they have translated it to a similarity measures as follow:

$$LCSim(C_1, C_2) = -log\left(\frac{length(path_{min}(C_1, C_2))}{2 \times D} \right) \quad (7)$$

Wu and Palmer (1994) define the similarity of two concepts based on their distance to the lowest common super concept and the distance of the common super concept to the root of taxonomy as well. The basic idea here is

that as the common subsumer goes far from the root the similarity of two concepts is become less sensitive to the distance of two concepts.

$$WuPalmerSim(C_1, C_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3} \tag{8}$$

where C_3 is common subsumer of C_1 and C_2. If C_1 and C_2 have more than one common subsumer, the C_3 would be the most specific one. N_1, N_2, and N_3 are the lengths of the paths between C_1 to C_3, C_2 to C_3, and C_3 to *root* respectively.

Information content based measures (Cross & Hu, 2011): Finding path between two words in a taxonomy graph is usually a time consuming task, so information content based measures just employ the content of two nodes to determine the similarity between their corresponding words. Taxonomy is enriched by an information content function as follows:

$$IC(c) = -log(p(c)) \tag{9}$$

IC represents the probability of encountering an instance of a concept c to all other concept instances.

Resnik (1995) have introduced the first measure based on the information content function.

$$ResnikSim(C_1, C_2) = max_{c \in S(C_1, C_2)}[IC(c)] \tag{10}$$

where $S(C_1, C_2)$ is the set of C_1 and C_2 common subsumers. This measure just considers the common subsumer which has the highest amount of information content.

Lin (1998) extends Resnik measure to also consider information content of C_1 and C_2 as well.

$$LinSim(C_1, C_2) = \frac{2 \times IC(C_3)}{IC(C_1) + IC(C_2)} \tag{11}$$

Jiang and Conrath (1997) have proposed another distance measure based on information content, but this measure has inspired by different idea. More data on common subsumer rather than on nodes themselves, the higher of the probability of similarity between the nodes.

$$JiangDistance(C_1, C_2) = IC(C_1) + IC(C_2) - 2 \times IC(C_3) \tag{12}$$

where C_3 is the common subsumer of C_1 and C_2 with the highest information content.

Combined measures: These measures utilize combinations of different measures. For example, it could exploit the positions of two nodes in taxonomy graph as well as their contents to find the similarity between two concepts.

Pirro (2009) has combined the idea of feature based similarity with the information content based measures to propose a new measure. Feature based similarity have been suggested by Tversky et al. (1977) employs the common features of C_1 and C_2 as well as their differentiating features as follows:

$$TverskySim(C_1, C_2) = \frac{|\, fe(C_1) \cap fe(C_2) \,|}{|\, fe(C_1) \cap fe(C_2) \,| + \alpha \,|\, fe(C_1) / fe(C_2) \,| + \beta \,|\, fe(C_2) / fe(C_1) \,|} \tag{13}$$

which fe is the feature set of the concept, and $\alpha, \beta \geq 0$ are the parameters of the Tversky similarity.

The Pirro similarity measure is defined as (Pirro, 2009):

$$PirroSim(C_1, C_2) = 3 \times IC(C_3) - IC(C_1) - IC(C_2) \tag{14}$$

where C_3 is the most informative common subsumer as defined in Resnik measure.

After briefly reviewing terminological similarity measures, we now discuss the terminological matching techniques used in ontology matching systems. Some of these ontology matching systems directly employ those similarity measures while some others use special techniques to perform the terminological matching.

OLA (Euzenat & Valtchev, 2003) utilizes a measure derived from Wu-Palmer for terminological mapping. ASMOV (Jean-Mary, Shironoshita, & Kabuka, 2009) uses Wu-Palmer to find similarity between properties while uses Lin measure to find similarity between concept names.

Having one of the highest results in the OAEI contest, Agreement Maker (Cruz, Antonelli, & Stroe, 2008) uses three different matchers for terminological matching. These matchers are Base Similarity Matcher (BSM), Parametric String-based Matcher (PSM), and the Vector-based Multi-word Matcher (VMM).

Agreement Maker uses BSM to generate the initial alignments among concept names of two ontologies. BSM first tokenizes the entity compound names and then removes all stop words (such as, "the", "a"). It then uses the WordNet to enrich each word in the tokenized string by its glossary. BSM then employs the stemming algorithm to translate the words of enriched strings to their roots. After these preprocessing steps, BSM uses following similarity measure to calculate the similarity between the two enriched concept names:

$$BaseSim(C_1, C_2) = \frac{2\times | D \cap D' |}{| D | + | D' |} \tag{15}$$

where D and D' are enriched versions of C_1 and C_2 respectively.

In PSM, users can choose various parameters which are suitable to the specific application. Users can select a set of string similarity measures such as Levenshtein or Jaro-Winkler, a set of preprocessing operations such as stemming or stop word elimination, and a set of weights for considered similarity measures. PSM computes the similarity between two concept names as the weighted average of values calculated by different selected similarity measures.

VMM enrich a concept name by extra information such as description field and neighbor names. Similarity between these enriched terms are then calculated using TF-IDF (Salton & McGill, 1986) technique.

RiMOM (Li, Tang, Li, & Luo, 2009) uses linear combination of modified Lin measure and a statistical measure. Falcon (Jian, Hu, Cheng, & Qu, 2005) employs a modified version of edit distance and combines the results by using the TF-IDF technique. CIDER (Gracia, Bernad, & Mena, 2011) uses Jaro Winkler measure to compute the similarity between concept names after enriching each concept name with its WordNet synonyms. CODI (Huber, Sztyler, & Noessner, 2011) combines various similarity measures such as Levenshtein and Jaro-Winkler through different methods. These methods include averaging, weighted averaging, maximizing. For weighted averaging methods, weights are calculated by a special learning method. AROMA (David, Guillet, & Briand, 2007) enhances the matching results by employing Jaro Winkler measure with a fixed threshold.

H-MATCH (Castano, Ferrara, & Montanelli, 2006) calculates the shortest path between the two entity names by using a thesaurus. LogoMap (Jimenez-Ruiz, Morant, & Grau, 2011) combines indexes, calculated by information retrieval technique, and anchor alignments to detect the matches among entities. LogoMap employs ISUB measure to compute anchor alignments confidences.

3. Approach

Within natural language, we use a vocabulary of atomic expressions and a grammar to construct well-formed and meaningful expressions as well as sentences. In the context of an ontology language the vocabulary is called *signature*, and can be defined as follows.

Definition 1 Signature. A signature $\mathbf{S}$ is a quadruple $\mathbf{S} = \langle \mathbf{C}, \mathbf{P}, \mathbf{R}, \mathbf{I} \rangle$ where $\mathbf{C}$ is a set of concept names, $\mathbf{P}$ is a set of object property names, $\mathbf{R}$ is a set of data property names, and $\mathbf{I}$ is a set of individual names. The union $\mathbf{P} \cup \mathbf{R}$ is referred to as the set of property names.

Definition 2 Similarity Search Algorithm σ . Given two ontologies O_1 and O_2 and their signatures $\mathbf{S}_1 = \langle \mathbf{C}_1, \mathbf{P}_1, \mathbf{R}_1, \mathbf{I}_1 \rangle$ and $\mathbf{S}_2 = \langle \mathbf{C}_2, \mathbf{P}_2, \mathbf{R}_2, \mathbf{I}_2 \rangle$, a similarity search algorithm σ is defined as:

$$\sigma(S, SimString) \rightarrow T \tag{15}$$

where $S = \mathbf{C}_2 \mid \mathbf{P}_2 \mid \mathbf{R}_2 \mid \mathbf{I}_2$ is the search space such that $T \in S$. $SimString \in \mathbf{S}_1$ is a search string. T type should be same as *SimString*, i.e. $SimString \in \mathbf{C}_1$ will lead to $T \in \mathbf{C}_2$ and so on. By reducing the problem with just considering one name from $\mathbf{S}_1$ as *SimString*, we tried to keep the algorithm more general, so it could be used by other applications such as search engines, which need to find a concept in an ontology similar to a search text. For the sake of the simplicity, in the followings, we only refer to concepts but similar methods could be applied to search in other parts of signatures.

In the following, we discuss the desired features for such similarity algorithm. First, the aim is to find the most specific concept in O_2 that is similar to *SimString*. This concept should not be more specific than *SimString*. Most of the semantic similarity measures that are defined based on edge counting are not applicable here since in such measures, concepts that have relation like sibling also receive a high similarity (due to close distance) and the direction of the relations does not matter. In fact, this requirement means that the algorithm should first try to find a concept very similar to *SimString* and if failed, should try to find a concept similar to more general meaning of *SimString*. In other words, if *SimStringConcept* be an ideational Concept in O_2 fully similar to *SimString*, then always $SimStringConcept \sqsubseteq foundConcept$. This feature is very important specially for aligning

two ontologies that expressed in different levels of granularity. Second, algorithm should not be sensitive to minor morphological differences such as suffixes and prefixes that have no effects on the meaning. In the same time, algorithm has to consider small none morphological changes that change the meaning totally (e.g., *Flood* and *Food*). Third, for some search applications such as search engines, high recall has a high importance, while for some other applications, like instance immigration between databases, more accurate results is definitely preferable. Consequently, the search algorithm has to be flexible and address different needs of recall and precision priority strategies. Fourth, the terminological search algorithm is the core component for most alignments systems, so having a high efficiency would directly improve the overall performance. Fifth, the algorithm should provide good level of stability because usually alignment algorithms are so sensitive to changes in their thresholds.

```
FINDSIMILAR(SimString,OntSearchList)
 1: ▷First tries to find resources that are similar to SimString directly
 2: SimOntRes ← FINDLEXICALSIMILAR(SimString,OntSearchList ,IsubThrshld )
 3: if SimOntRes = NIL then
 4:    if SEMANTICFILTERACCEPTS(SimOntRes.LocalName,SimString) then
 5:       return SimOntRes
 6:    end if
 7: end if
 8: ▷CreatingSearchMatrix
 9: M ← WORDNETNUMBEROFMEANING(SimString)
10: SimMatrix ← BUILDEMPTYSIMILARITYMATRIX(M)
11: for i ← 0 to M − 1 do
12:    ADDTOROW(SimMatrix,i,WORDNETGETSYNONYMS(SimString,i))
13:    APPENDTOROW(SimMatrix,i,WORDNETGETHYPERNYMS(SimString,i))
14: end for
15: ▷CalculateMostSimilars
16: CALCULATESIMILARITIES(OntSearchList , SearhMatrix)
17: CandidateArray ← BUILDARRAY(M)
18: for i ← 0 to M − 1 do
19:    CandidateArray [i] ← FINDCANDIDATE( SearhMatrix ,i)
20: end for
21: ▷ WordSenseDisambiguation
22: preferredMeaning ← WSD(SearchMatrix[i])
23: if CandidateArray[preferredMeaning][i] = NIL then
24:    return CandidateArray[preferredMeaning].MostSimilarOntRes
25: end if
26: ▷If WSD failed
27: for i ← 0 to M − 1 do
28:    if CandidateArray [i] = NIL then
29:       return CandidateArray[i].MostSimilarOntRes
30:    end if
31: end for
32: ▷ Not found
33: return NIL
```

Figure 1. Similarity search algorithm

Proposed similarity search algorithm has been expressed in pseudo-code in Figure 1. It first tries to find the concepts in *OntSearchList* that are lexically very similar to *SimString*. This step is carried out by using a lexical search algorithm depicted in Figure 3. FindLexicalSimilar compares *SimString* to the all names exist in *OntSearchList*. Once it found the concept name that has the highest similarity to *SimString*, it compares their similarity value to a threshold, and if the similarity surpasses the needed threshold, it will return found similar concept. Otherwise, it will return null. For this direct comparisons, algorithm uses ISUB similarity measure

which, as reported (Stoilos et al., 2005), has a clear superiority in ontology name searching over other lexical measures. Once the lexical algorithm finished, the algorithm checks the returned concept (if exists any returned similar concept) by a semantic filtering algorithm (see Figure 4) to ensure that there is no semantic inconsistency between it and *SimString*.

As aforementioned in requirements, proposed algorithm should prevent the cases that two concepts are very similar lexically, but quite different semantically. We have added a SemanticFilteringAccepts method to fulfill this requirement. This method calculates the semantic similarity of two concepts by JIANG measure. If the similarity could not be determined, this method has not enough information to reject the similarity (showed in method by -1). If the proposed similar concepts have a semantic similarity lower than a threshold, Semantic Filtering Accepts will reject their similarity. Otherwise, the method accepts their similarity.

In the case that the algorithm fails to find a direct similar concept, it tries to find similar concepts by extending the *SimString* (lines 8-14). This algorithm extracts all *SimString* synonyms and hypernyms from WordNet and then creates matrix illustrated in Figure 2. Each row of this matrix represents one meaning of *SimString*. In fact, this algorithm not only tries to find concepts similar to *SimString*, but also tries to find concepts similar to synonyms and hypernyms of *SimString* as well. In each row of this matrix, all synonyms of the meaning that row represents, come first (from left) and all their hypernyms from most specific to most general come afterwards.

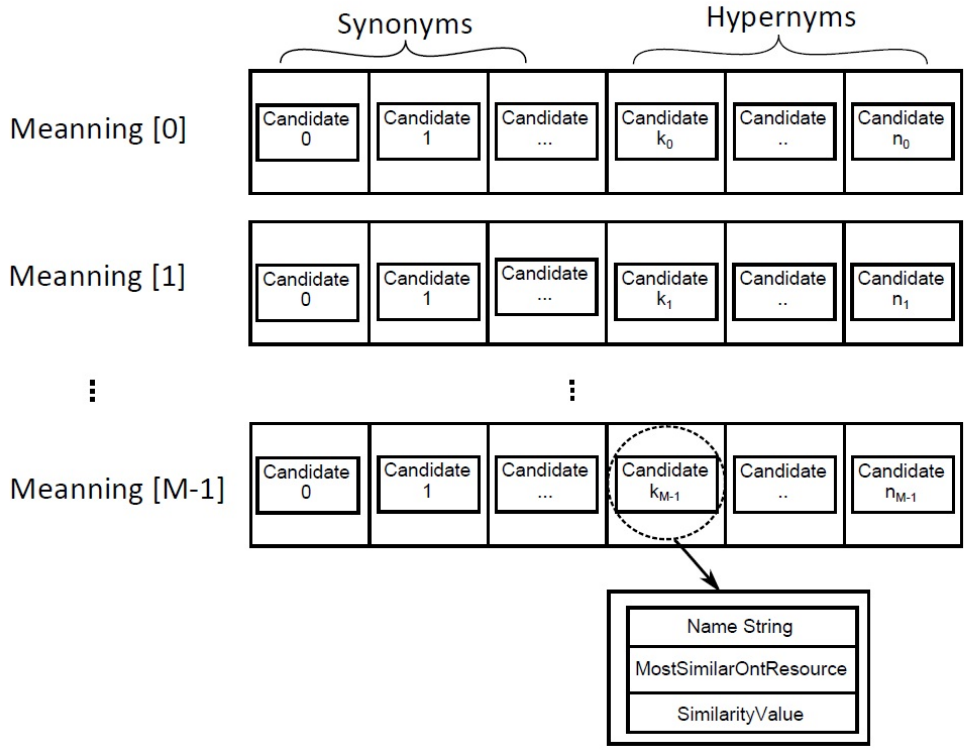

Figure 2. Searching matrix structure

Each *SimMatrix[i][j]* from this matrix can store one candidate for search result and contains three data fields; a *NameString* which algorithm tries to find most similar concept in *OntSearchList* to this and store it in *MostSimilarOntRes*, *MostSimilarOntRes* stores most similar concept to *NameString* which has been found so far, and finally, a *SimilarityValue* which shows the degree of similarity between *NameString* and *MostSimilarOntRes*.

CalculateSimilarites method (see Figure 5) compares each concept name from *OntSearchList* to all *NameString*s in matrix; if the concept name similarity to *NameString* is more than *SimilarityValue*, *MostSimilarOntRes* is replaced by this new concept, and *SimilarityValue* is also updated. Once all ontology concepts were compared to matrix candidates, each *SimMatrix[i][j]* would contain ontology concept that is most similar to containing *NameString*. For comparing the extracted synonyms and hypernyms to ontology concepts, we simply use the Levenshtein similarity measure since we know that ISUB has failed in previous step, and we want to use an alternative measure.

```
FINDLEXICALSIMILAR(SimString,OntSearchList,ThrshId)
 1: SimOntRes ← NIL
 2: SimOntResSimValue ← 0.0
 3: for all OntRes ∈OntSearchList do
 4:    Name ← OntRes · LocalName
 5:    Sim ← ISUBSIMILARITY(SimString,Name)
 6:    if Sim > SimOntResSimValue then
 7:       SimOntRes ← OntRes
 8:       OntResSimValue ← Sim
 9:    end if
10: end for
11: if Sim > ThrshId then
12:    return SimOntRes
13: end if
14: return Nil
```

Figure 3. Find lexical similar algorithm

As noted earlier, each row of matrix includes all synonyms and hypernyms of one meaning of *SimString*, in order which most specific names at left to most general at right. By considering this order, a *DistanceThreshold*, values between [0, 1], specifies the ratio of each row from left that algorithm is permitted to scan for finding similar concepts. In other words, lower *Distance Threshold* means to consider less general hypernyms. Conversely, higher *Distance Threshold* means to consider more general hypernyms. Using this parameter can provide more flexibility which specified in requirements.

Following this, algorithm chooses at most one candidate from each row of this matrix by calling FindCandidate (see Figure 5) and puts them in *CandidateArray*. In the Find Candidate one row of matrix, which represents one meaning of *SimString,* is scanned from left to right to find the first candidate that has a Similarity Value higher than SimThreshold. If found candidate is rejected by Semantic Filter Accepts method, the selected candidate is simply ignored and scanning in the same row will be continued to find another possible candidate.

```
SEMANTICFILTERINGACCEPTS(OntResName,SimString)
1: Sim ← JIANGSIMILARITY(OntResName,SimString)
2: if Sim = −1 or Sim > SemanticFilteringLowThreshold then
3:    return true
4: else
5:    return false
6: end if
```

Figure 4. Semantic filtering accepts algorithm

Once the algorithm finished establishing the possible candidate list, it can return the candidate list to let the application to choose the suitable candidate upon its preferred context. Alternatively, it can do the word sense disambiguation (WSD) (Navigli, 2009) by itself and then return the selected candidate. In this algorithm we have implemented a straightforward WSD technique using context knowledge which has been accumulated in *SearchMatrix*. Each row of *SearchMatrix* contains synonyms and hypernyms of one *SimString*'s meaning. As aforementioned, after calling the CalculateSimilarites method, each element of *SearchMatrix* is populated by the most similar concept in addition to the similarity value. The WSD method uses average of similarity values in each row as measure of its relatedness to the ontology. The higher row's similarity values average, the higher probability that the row represents the wanted meaning of *SimString* in ontology. The rationale behind this rule is that having higher average shows that the row has more commonality with the ontology concept names.

```
CALCULATESIMILARITIES(OntSearchList,SearchMatrix)
 1: for all OntRes ∈ OntSearchList do
 2:   for i ← 0 to M − 1 do
 3:     for j ← 0 to ROWSIZE(SimMatrix, i) × DistanceThreshold do
 4:       Sim ← LEVENSIMILARITY(Name,SimMatrix[i][j] · NameString )
 5:       if Sim > SimMatrix[i][j] · SimilarityValue then
 6:         SimMatrix[i][j] · SimilarityValue ← Sim
 7:         SimMatrix[i][j] · MostSimilarOntRes ← OntRes
 8:       end if
 9:     end for
10:   end for
11: end for
```

Figure 5. Calculate similarities algorithm

Finally, the algorithm will return the selected candidates, and if the WSD fails to choose any candidate (i.e. the selected row does not contain any candidate), the algorithm tries to select the first candidate based on their order in the WordNet because the synsets in the WordNet are sorted by their usages' frequencies (Fellbaum, 1998).

```
FINDCANDIDATE(SearchMatrix, Row, ThrshId, SimString)
 1: for j ← 0 to ROWSIZE(SearchMatrix, i) × DistanceThreshold do
 2:   if SearchMatrix [Row][j] · SimilarityValue > LevThrshId then
 3:     OntResName ← SearchMatrix[Row][j].MostSimilarOntRes.LocalName
 4:     if SEMANTICFILTERINGACCEPTS(OntResName ,SimString) then
 5:       return SearchMatrix[Row][j]
 6:     end if
 7:   end if
 8: end for
 9: return Nil
```

Figure 6. Find candidate algorithm

4. Experiments

For examining performance of implemented search function, OAEI benchmark (Euzenat et al., 2011) 101 and 205 datasets have been used to compare suggested search algorithm to different search algorithms based on variety of syntactical and semantical similarity measures. OAEI benchmark 205 dataset has been designed to show effectiveness of ontology matching algorithms in using string similarity. In our proposed algorithm, three different syntactical and semantical measures have been used: the ISUB measure as the main syntactical measure, the Levenshtein as the auxiliary syntactical measure, and the JIANG as the semantical measure for filtering mechanism. In our test scenarios we compared results of our approach to use of these three measures separately. In addition, to investigate performance of our algorithm more comprehensively, some other similarity measures from different group of semantic measures have been exploited in our test scenarios. These similarity measures include Lin, Resnik, and Pirro. It has to be noted that we cannot compare results of our algorithm, which just use name similarity, with full ontology matching algorithms that use many various matching algorithm.

Furthermore, we carried out two extra tests that consider aggregation of these measures as well. In the first aggregation scenario, the average of all these measures is been calculated while, in the second scenario, these three measures are prioritized according to their order in our algorithm. We also added some more experiences in order to investigate some other aspects of proposed algorithm such as its stability and effectiveness of features like semantic filtering or word sense disambiguation.

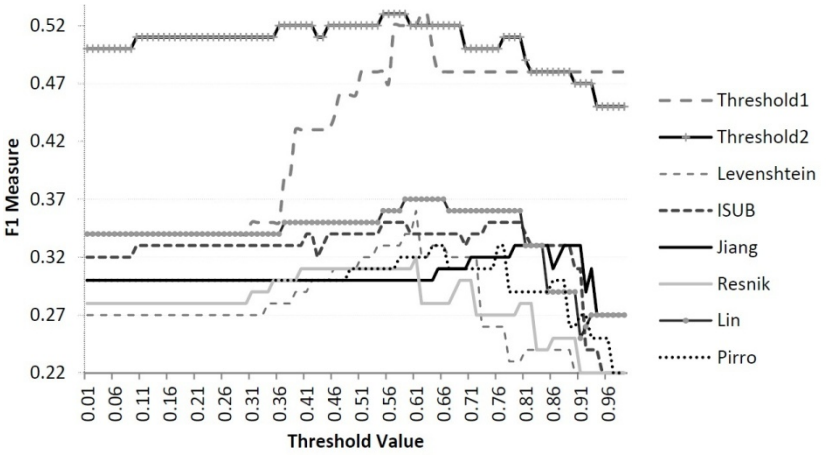

Figure 7. F1 measures of achieved results have been compared to other search algorithms based on well-known semantic and string similarity measures

Description: Threshold1 shows the results which has been achieved by changing the first threshold of algorithm, and Threshold 2 shows the results for changing second threshold of algorithm.

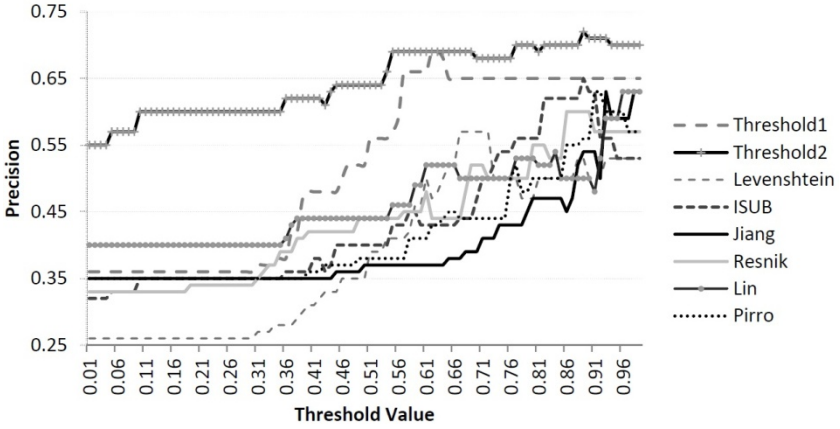

Figure 8. Precisions of achieved results have been compared to other search algorithms based on other well-known semantic and string similarity measures

The proposed algorithm, like most of the other matching algorithms, needs some setting parameters. First, it needs the threshold used for the main lexical similarity search algorithm (see Figure 3) which uses the ISUB similarity measure, so we will refer to it as ISUB threshold or sometimes simply threshold1. Second, it requires the threshold for semantic filtering (see Figure 4) which uses the JIANG similarity measure. Third, it exploits a threshold for finding candidates for each row (see Figure 6). Since this algorithm uses Levenshtein measure, we refer to this threshold as Levenshtein threshold. Finally, there is also a distance threshold which put a limitation on the percentage of hypernyms which considered in each row of the *Search Matrix*. We focus more on the first and second thresholds, but we also accomplish some experiences on the effects of changing other thresholds on the algorithm performance.

In Figure 7, Figure 8, and Figure 9, performance of proposed algorithm has been compared with the matchers have made based on other measures in the term of F1 measure, precision, and recall respectively. For each of these measures, we have developed a basic matcher and applied it to the data set. In each matcher, regarding threshold was set from 0.0 to 0.99 with the step of 0.01. For our algorithm, the *Thrshold 1* shows the results of changing ISUB threshold, and the *Thrshold 2* shows the effects of changing in Levenshtein threshold while keeping the other thresholds in the fixed best values. These results show that our algorithm outperforms all other measures when they are used separately in the term of result quality factors. These results also reveal that the algorithm is more sensitive to changes in the first threshold other than changes in the second threshold.

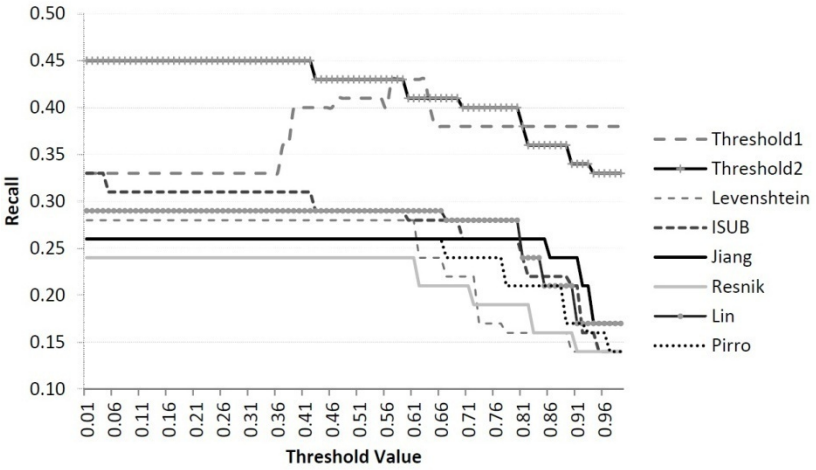

Figure 9. Recall results of proposed method have been compared to other search algorithms based on other well-known semantic and string similarity measures

The previous experiment has been repeated to compare the run time needed for each developed matcher. Although the developed algorithm running time is not comparable with syntactical measures such as Levenshtein (run time was less than 20 MS and has not been shown), the Figure 10 shows that it has a running time far better than other semantic measures.

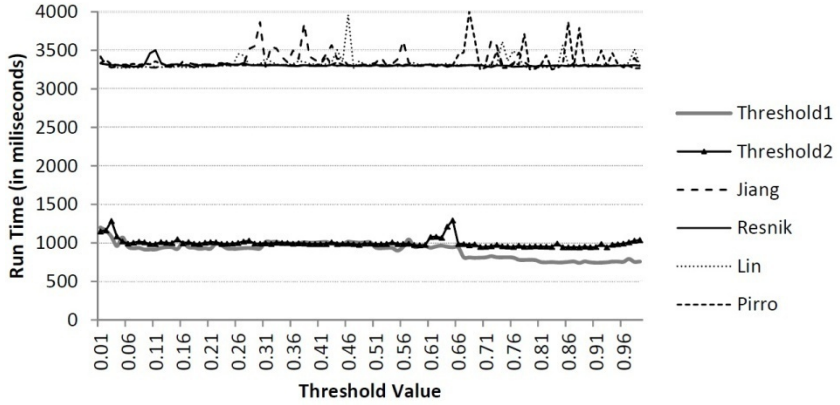

Figure 10. Run time comparison of proposed algorithm to other search algorithms based on well-known semantic measures

The other concern about the using of an algorithm with numbers of thresholds is the stability of the algorithm. In other words, algorithm has to show that its performance would not change sharply by changing its thresholds. Figure 11 shows the changes in F1 measure of the results due to changes in both first and second thresholds. These results demonstrate that algorithm is not so sensitive to its thresholds changes. It shows a good level of stability because the majority area of the curve has a high F1 measure, and the curve is smooth without sharp drops or raises.

One feature of the proposed algorithm is the using of a *distance threshold* to put a limit on the percentage of used hypernyms in each line of search matrix (see Section 3). Figure 12 illustrates the effects of changing *distance threshold* from 0.0 to 1.0 while all other parameters such as similarity and filtering thresholds are in their best values. This results show that after a fluctuation, F1 measure has been increased to its highest value when the *distance threshold* is 0.81 and then has been decreased again afterward. This experience shows that using more hypernyms will improve the results, but employing very general hypernyms shows fewer impacts on the results.

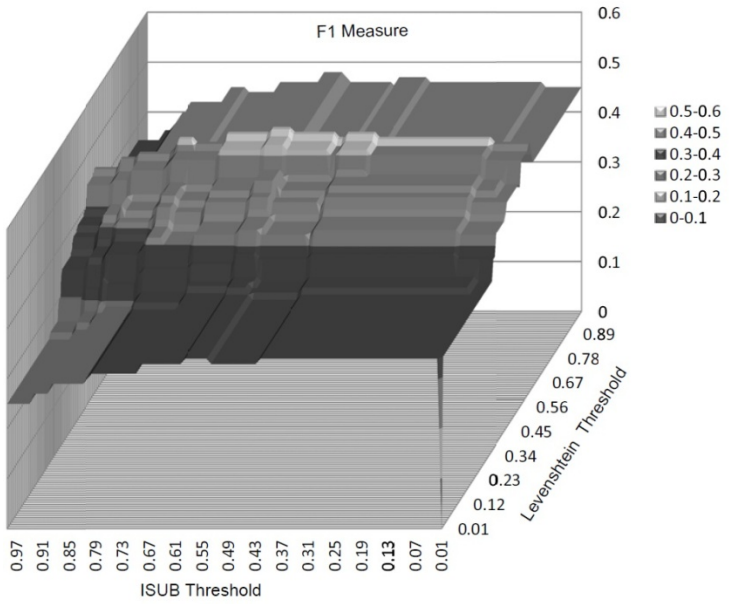

Figure 11. Searching for best combinations of both thresholds of algorithm

Description: ISUB axis represents the values for the first threshold of algorithm and the Levenshtein axis shows the values for the second threshold of algorithm.

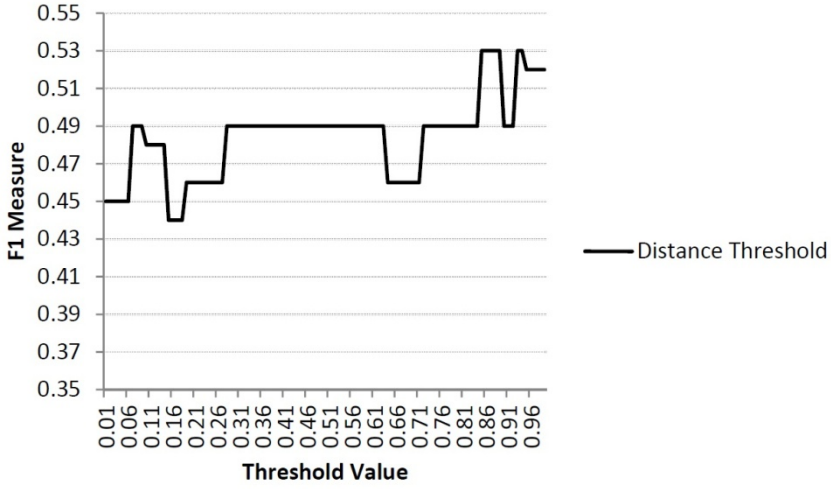

Figure 12. The effect of changing distance threshold parameter on the results of algorithm in the term of F1 measure

As discussed earlier (see Section 3), this algorithm applies a filtering mechanism to eliminate the false similar causes that the two words are lexically very similar but semantically different. Figure 13 shows the significant improvement which has been achieved by employing this mechanism independent from the used similarity thresholds. In this experiment, we changed the first ISUB threshold from 0.0 to 1.0. However, the runtime values illustrated in the Figure 14 show that this mechanism has made the algorithm almost three times slower.

Figure 15 demonstrates that applying the simple proposed word sense disambiguation mechanism can improve the results. Additionally, as illustrated in Figure 14, the word sense disambiguation due to its simplicity will not put significant overload on the main algorithm.

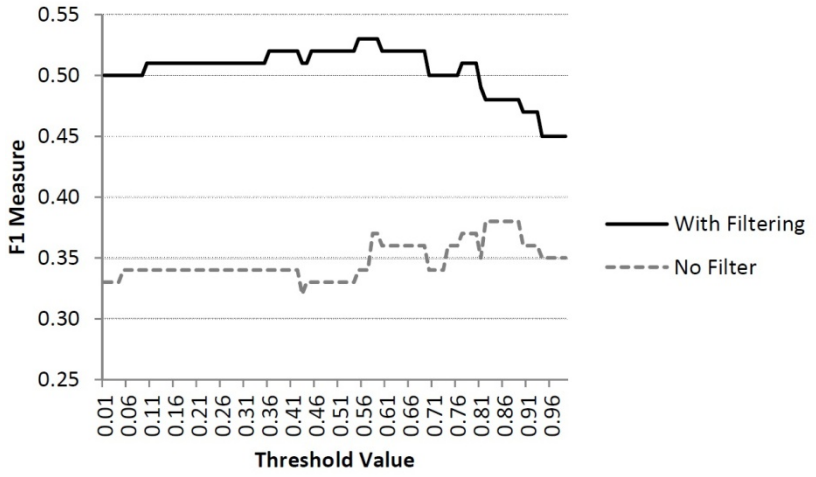

Figure 13. The achieved results of algorithm which uses filtering mechanism compared to results achieved without using filtering

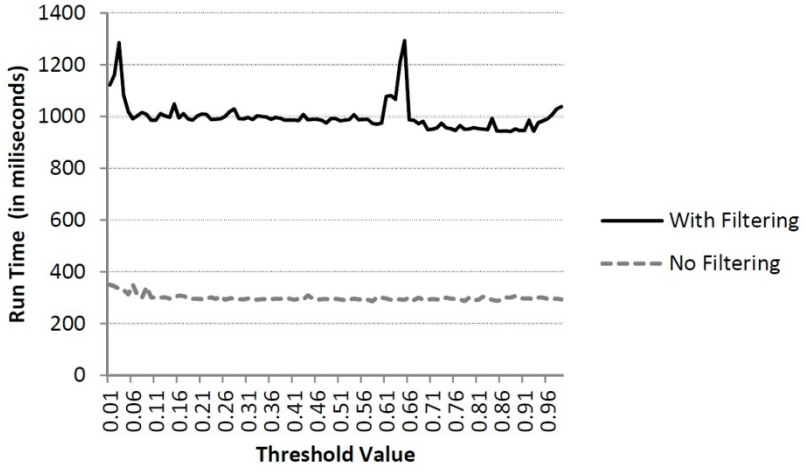

Figure 14. The cost of using filtering mechanism has been showed by comparing run time of algorithm that uses filtering mechanism and the run time of algorithm that does not use filtering

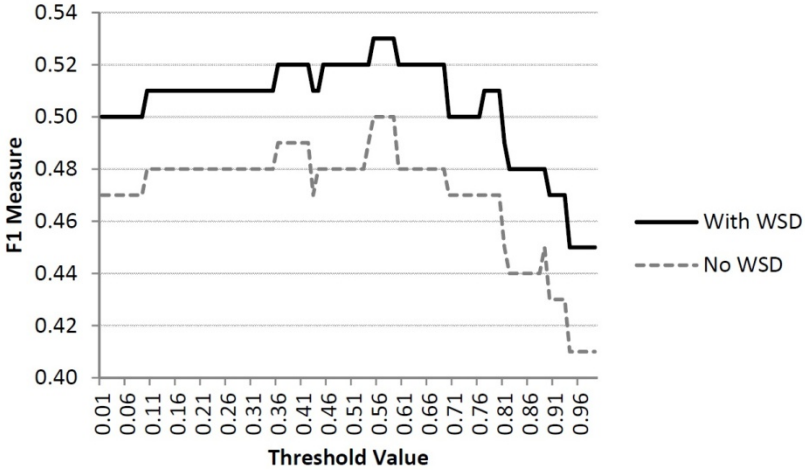

Figure 15. Results of algorithm which use word sense disambiguation mechanism compared to results has been achieved without using word sense disambiguation

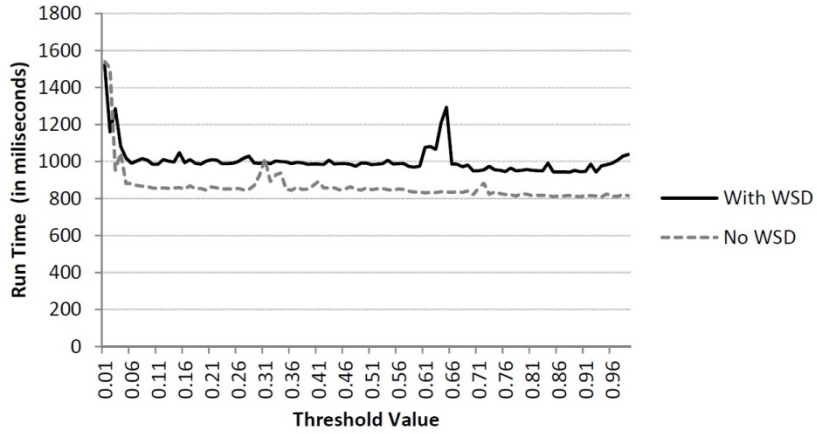

Figure 16. The cost of using WSD mechanism has been showed by comparing run time of algorithm that uses WSD mechanism with run time of the algorithm does not use WSD

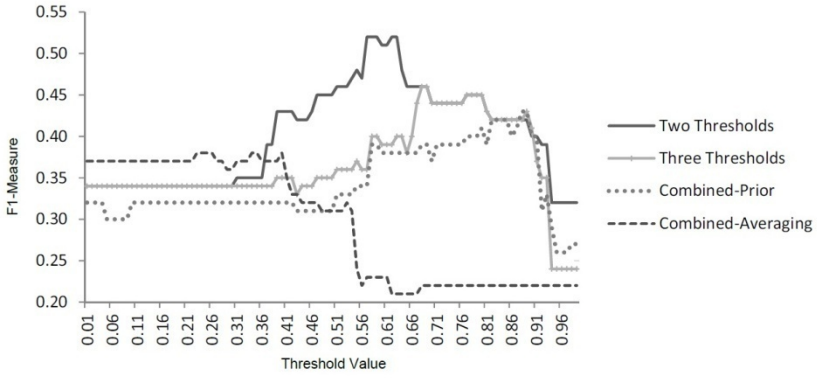

Figure 17. The results of proposed algorithm compared to other algorithms that combine different measures

Our algorithm uses three measures which are Isub, Levenshtein, and JIANG. In the last experience, it has been tried to compare suggested algorithm (*Three Threshold* in Figure 17) to some other previous methods that also combine those measures to improve the overall performance. Two conventional approaches are: comparing average of all three different measures (*Combined-Averaging* in Figure 17) and using different measure with priority (*Combined-Prior* in Figure 17). In using with priority case, it begins to calculate the similarity with the first measure and then compares it to the threshold. If the similarity value is less than the threshold, it will use the next measure. In the other hand, this method considers that two words are similar if at least one calculated similarity is higher than the threshold. It should be noted that the aforementioned priority just has influence on the algorithm runtime and not the results. Setting some thresholds to the best values in previous experiments can be subject of argumentation, so it has been avoided to use any specific values for the needed thresholds in this last experiment. In the configuration of our algorithm's thresholds, instead of using fixed value for each threshold, we have just used a same variable value for all needed thresholds.

Figure 17 compares the results for those three different approaches (*Two Threshold* results are the results achieved by a fixed value for semantic filtering mechanism and could be used as an estimation of proposed algorithm final results). This figure shows that in the first, the average approach demonstrates a better performance, but with increase in the threshold this method performance has been dropped to the lowest values such that it is not comparable to other methods. Priority based approach has also demonstrated lower F1 measure in comparison with proposed algorithm. This experiment shows that even without taking in consideration any threshold setting the proposed algorithm has surpassed its competing approaches.

5. Conclusions

In this paper, we proposed a novel terminological search algorithm which tries to find a concept Name (property or individual) similar to an input Search String in a given ontology. This search algorithm is a basic building

block for many semantic applications such as ontology matching systems and ontology search engines. As we showed in related works, while there exist a lot of ontology matching approaches and also many proposed similarity measures, little attentions have been paid to develop similarity search algorithms that exploit different similarity measures. Such algorithm can use various similarity measures to sum up their advantages and reduce effects of their weakness.

Our suggested algorithm extends the input search string by creating a matrix from its synonym and hypernyms which have been extracted recursively from WordNet such that each row of this matrix represents one meaning. Each row includes synonym from left and then most specific hypernyms and finally more general hypernyms come afterwards. The algorithm first uses the ISub measure to find similar concepts, and if failed to find any lexically similar concept, it will continue to search the extension matrix. In both cases, algorithm uses JIANG semantic similarity measure to detect wrong candidates which are lexically similar and semantically different. For coping with the word polysemy problem, algorithm use a simple word sense disambiguation method based on average relatedness of each row of the search matrix. The algorithm employs and combines different kind of similarity measures in different situations to achieve a higher performance, accuracy, and stability compared to previous methods which either use one similarity measure or combine them in a naive ways such as averaging.

For algorithm evaluation, we used OAEI Bench mark data set and the achieved results showed the superiority of proposed algorithm and effectiveness of different suggested mechanism such as word sense disambiguation and semantic filtering mechanism.

There are some potential limitations in this study. First, the proposed algorithm is not well suited for search names that have many parts since it does not employ any tokenization method. This is an important requirement which should be implemented before embed this algorithm in a real ontology matching system because some major ontologies especially from medical world usually use long concept names. Nonetheless, in some applications such as semantic search engines this search algorithm in its current implementation could be very useful. It should be noted that in complex matching task (Ritze, Volker, Meilicke, & Sváb-Zamazal, 2010) usually finding concepts similar to a part of a compound name is an important basic operation.

Second, this algorithm mainly relies on the WordNet to enrich the search string while ontologies are defined in different domains and need taxonomy and background knowledge that better fit their requirements. This algorithm also could exploit the content of each concept in implementing task such as WSD or semantic filtering.

In future works, we will broaden our approach to use more linguistics techniques such as tokenization, stop word reduction, and stemming to make it more suitable to cope with ontologies that have long compound names. We would also like to generalize algorithm such that employing knowledge sources other than WordNet become possible in different situations. Specially, employing ontologies themselves in building extension matrix could be an interesting improvement. Another future research direction is to exploit more sophisticated and state of the art word sense disambiguation techniques.

References

Berners-Lee, T., Hendler, T., & Lassila, J. (2001). The Semantic Web. *Scientific American*. http://dx.doi.org/10.1038/scientificamerican0501-34

Castano, S., Ferrara, A., & Montanelli, S. (2006). Matching Ontologies in Open Networked Systems: Techniques and Applications. *J. Data Semantics V.,* 25-63.

Cross, V., & Hu, X. (2011). Using Semantic Similarity in Ontology Alignment.

Cruz, I. F., Antonelli, F. P., & Stroe, C. (2008). *Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods*. Paper presented at the OM.

David, J., Guillet, F., & Briand, H. (2007). Association Rule Ontology Matching Approach. *International Journal of Semantic Web Information Systems, 3*(2), 27-49. http://dx.doi.org/10.4018/jswis.2007040102

Department, Z. W., & Wu, Z. (1994). Verb Semantics And Lexical Selection. *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 133-138.

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & dos Santos, C. T. (2011). Ontology Alignment Evaluation Initiative: Six Years of Experience. *J. Data Semantics, 15*, 158-192.

Euzenat, J., & Shvaiko, P. (2007). *Ontology matching*. Springer-Verlag.

Euzenat, J., & Valtchev, P. (2003). *An integrative proximity measure for ontology alignment*. Paper presented at

the The 1st Intl. Workshop on Semantic Integration.

Fellbaum, C. (1998). WordNet: An Electronic Lexical Database.

Gracia, J., Bernad, J., & Mena, E. (2011). *Ontology Matching with CIDER:evaluation report for OAEI 2011*. Paper presented at the The 6th International Workshop on Ontology Matching.

Huber, J., Sztyler, T., & Noessner, J. (2011). *CODI: Combinatorial Optimization for Data Integration*. Paper presented at the The 6th International Workshop on Ontology Matching.

Jaro, M. A. (1995). Probabilistic Linkage of Large Public Health Data Files. *Statistics in Medicine, 14*, 491-498. http://dx.doi.org/10.1002/sim.4780140510

Jean-Mary, Y. R., Shironoshita, E. P., & Kabuka, M. R. (2009). Ontology matching with semantic verification. *J. Web Sem., 7*(3), 235-251. http://dx.doi.org/10.1016/j.websem.2009.04.001

Jian, N., Hu, W., Cheng, G., & Qu, Y. (2005). *FalconAO: Aligning Ontologies with Falcon.* Paper presented at the Integrating Ontologies.

Jiang, J., & Conrath, D. (1997). *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy*. Paper presented at the The International Conference Research on Computational Linguistics (ROCLING).

Jimenez-Ruiz, E., Morant, A., & Grau, B. C. (2011). LogMap results for OAEI 2011.

Leacock, C., & Chodorow, M. (1998). *Combining local context and WordNet similarity for word sense identification*. Paper presented at the MIT Press.

Levenshtein, V. I. (1966). *Binary codes capable of correcting deletions, insertions, and reversals.* Paper presented at the Soviet Physics Doklady.

Li, J., Tang, J., Li, Y., & Luo, Q. (2009). RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering, 21*, 1218-1232.

Lin, D. (1998). *An Information-Theoretic Definition of Similarity*. Paper presented at the ICML.

Navigli, R. (2009). Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS, 41*(2), 1-69. http://dx.doi.org/10.1145/1459352.1459355

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology, 48*(3), 443-453. http://dx.doi.org/10.1016/0022-2836(70)90057-4

Pirro, G. (2009). A semantic similarity metric combining features and intrinsic information content. *Data Knowl. Eng., 68*, 1289-1308. http://dx.doi.org/10.1016/j.datak.2009.06.008

Rada, R., Mili, H., Bicknell, E., & Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics, 19*(1), 17-30. http://dx.doi.org/10.1109/21.24528

Resnik, P. (1995). *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*. Paper presented at the In Proceedings of the 14th International Joint Conference on Artificial Intelligence.

Ritze, D., Volker, J., Meilicke, C., & Sváb-Zamazal, O. (2010). *Linguistic Analysis for Complex Ontology Matching*. Paper presented at the Proceedings of the 5th International Workshop on Ontology Matching (OM-2010), Shanghai, China, November 7, 2010.

Romero, M. M., Vázquez-Naya, J. M., Loureiro, J. P., & Ezquerra, N. (2009). Ontology Alignment Techniques *Encyclopedia of Artificial Intelligence* (pp. 1290-1295): IGI Global.

Salton, G., & McGill, M. J. (1986). *Introduction to Modern Information Retrieval*: McGraw-Hill, Inc.

Stoilos, G., Stamou, G., & Kollias, S. (2005). A String Metric for Ontology Alignment. *The Semantic Web – ISWC 2005, 3729*, 624-637.

Tversky, A. (1977). *Features of Similarity*. Paper presented at the Psychological Review.

Winkler, W. E. (1999). *The state of record linkage and current research problems*.