Ontology Matching with Word Embeddings

Yuanzhe Zhang[1], Xuepeng Wang[1], Siwei Lai[1], Shizhu He[1], Kang Liu[1], Jun Zhao[1], and Xueqiang Lv[2]

[1] National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{yzzhang,xpwang,swlai,shizhu.he,kliu,jzhao}@nlpr.ia.ac.cn
[2] Beijing Key Laboratory of Internet Culture and Digital Dissemination Research
Beijing Information Science & Technology University
lxq@bistu.edu.cn

Abstract. Ontology matching is one of the most important work to achieve the goal of the semantic web. To fulfill this task, element-level matching is an indispensable step to obtain the fundamental alignment. In element-level matching process, previous work generally utilizes WordNet to compute the semantic similarities among elements, but WordNet is limited by its coverage. In this paper, we introduce word embeddings to the field of ontology matching. We testified the superiority of word embeddings and presented a hybrid method to incorporate word embeddings into the computation of the semantic similarities among elements. We performed the experiments on the OAEI benchmark, conference track and real-world ontologies. The experimental results show that in element-level matching, word embeddings could achieve better performance than previous methods.

Keywords: Ontology Matching, Element-level Matching, WordNet Similarity, Latent Semantic Analysis, Word Embeddings.

1 Introduction

The semantic web is receiving increasing attentions because of its bright future. In the semantic web, ontologies are essential components which are explicit specifications of conceptualization [10]. Thus, a large number of ontologies have been created in the last decade. Although many of them describe the same domain, they cannot share information with each other since they are designed by different conventions. Hence, ontology matching is required due to the heterogeneous and distributed nature of ontologies [4].

Many ontology matching systems have been designed in recent years. Element-level techniques [7] are widely utilized. They take advantage of lexical information as essential elements. However, merely employing the string surface similarity is impracticable. For example, "journal" and "periodical" cannot be matched, while "journal" and "journey" actually might be aligned. To settle this problem, WordNet similarity [22] was employed to obtain the semantic similarities among elements. But the weakness is that the words in WordNet are insufficient. Many elements in ontologies cannot find their correspondences in WordNet. As a result, it is impossible to compute the semantic similarities between these elements with others.

M. Sun et al. (Eds.): CCL and NLP-NABD 2014, LNAI 8801, pp. 34–45, 2014.

We aim to acquire the semantic similarity without the scale constraint of WordNet. So we introduce word embeddings to this task. Word embeddings are able to represent a majority of words as vectors in a semantic space. The words' similarities then can be worked out by using simple strategies, such as the cosine similarity, Euclidean distance, etc. Word embeddings have been proved to be effective in several NLP tasks, such as named entity recognition, part-of-speech tagging and semantic role labeling [5]. However, there is no straightforward evidence that word embeddings are effective for ontology matching. Thus, in this paper, we learn word embeddings using Wikipedia as training data, and testify the capacity of them.

Compared with edit distance, word embeddings can deeply capture the real meaning behind the words. Compared with WordNet, word embeddings already contain more words. Moreover, we proposed a hybrid method to combine word embeddings and edit distance together. To the best of our knowledge, this is the first usage of word embeddings in ontology matching field.

We conducted several experiments on several open datasets, such as Ontology Alignment Evaluation Initiative (OAEI) 2013[1] benchmark and conference track, real-world ontologies (Freebase [2], YAGO2 [11] and DBpedia [1]). Concretely, we compared the performance of edit distance, WordNet, Latent Semantic Analysis (LSA), word embeddings and a hybrid method using both edit distance and word embeddings. The results demonstrate that the performance of the hybrid method outperforms the others.

The rest of this paper is organized as follows. Section 2 reviews related work. In Section 3, we describe the setup of the paper. Section 4 presents our matching algorithm in detail. Section 5 shows the experiments and the analysis. We conclude this paper in the final section.

2 Related Work

Ontology matching, a solution to the semantic heterogeneity problem [23], is a crucial part to achieve the goal of the semantic web. So far, dozens of ontology matching systems have been created. Lexical information, including names and labels describing entities are valuable to matching systems. Essential correspondences between two ontologies are found by element-level matchers. A simple and efficient method is to calculate the surface similarity between the strings. Most of the string-based metrics have been evaluated [25,3]. Edit distance was widely adopted by many matching systems, such as RiMOM [14], ASMOV [12] and AgreementMaker [6], to measure the similarity between two words.

It is evident that string surface similarity fails to capture the semantics behind the strings, like "journal" and "periodical". To deal with this problem, WordNet [20] was employed for ontology matching. Most of the state of the art matching systems took advantage of WordNet [17]. There are three principal ways to obtain WordNet similarity [17], namely edge-based method, information based method and hybrid method. WordNet definitely discovers the meaning of a word and resolves part of the synonym problem. The biggest shortcoming of WordNet is its low coverage. If either of the two words is out of range, the WordNet similarity between them cannot be figured out.

[1] http://oaei.ontologymatching.org/2013/

Distributed word representations are called word embeddings which are trained through deep neural networks. Each dimension of the embeddings represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties [26].

Word embeddings have been used in many NLP tasks and have received a number of positive results. These tasks include parsing [24], language modeling [19] and sentiment classification [9], etc. Collobert et al. [5] proposed a unified neural network architecture and learning algorithm, which improved the performance of named entity recognition, part-of-speech tagging and chunking. However, to the best of our knowledge, there is little work which employed word embedding in the task of ontology matching.

3 Setup

3.1 Problem Statement

We define an ontology O by a simplified 3-tuple (C, P, I). C stands for the *classes*, denoting the concepts. P represents the *properties*, indicating the relations within the ontology. I means the *individuals*, which are the instances of classes. Classes, properties and individuals are collectively referred to as e, *entities*. Entities have several lexical descriptions, i.e., *names*,*labels* and *comments*. For instance, an entity's name is "Journal";its label is "Journal or magazine" and its comment is "A periodical publication collecting works from different authors".

The task of ontology matching is to find the alignment between entities in a source ontology O_1 and a target ontology O_2. We define an alignment as a set, $\{(e_1, e_2, con)|e_1 \in O_1, e_2 \in O_2\}$. Every element in the set is called a correspondence. e_1 is an entity in O_1, and e_2 is an entity in O_2. con is the confidence of the correspondence.

Our algorithm offers an implement of an element-level matcher. We declare two preconditions in advance. First, the hierarchy structure is not taken into account. It is because the structure-level matchers are error-prone and strongly depend on the results of element-level matchers [21]. Only contrasting the performance of the element-level matchers makes it easier to discover the capability of the newly added word embeddings method. Second, following the convention of OAEI 2013 benchmark and conference track, the correspondences only have equivalence ($\equiv$) relations.

3.2 WordNet Similarity

There are many proposed ways to use WordNet to acquire word similarity. In this paper, we employ three representative methods.

Edge-based method. Wu and Palmer [27] define the similarity between two concepts taking advantage of common super-concept and paths.

$$Sim(C_1, C_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3} \tag{1}$$

Where C_3 is the least common superconcept of C_1 and C_2. N_1 is the number of nodes on the path from C_1 to C_3. N_2 is the number of nodes on the path from C_2 to C_3. N_3 is the number of nodes on the path from C_3 to root.

Information-based method. Lin [16] presents an information-based similarity computational method. It bases on the idea that the similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are. The final formula is as follows.

$$Sim(x_1, x_2) = \frac{2 \times \log P(C_0)}{\log P(C_1) + \log P(C_2)} \tag{2}$$

Where $x_1 \in C_1$ and $x_2 \in C_2$. C_0 is the most specific class that subsumes both C_1 and C_2.

Hybrid method. Jiang and Conrath [13] propose a combined model, adding information content to edge-based method to measure the semantic similarity between words.

$$Dist(w_1, w_2) = IC(c_1) + IC(c_2) - 2 \times IC(LSuper(c_1, c_2)) \tag{3}$$

where $c_1 = sen(w_1)$ and $c_2 = sen(w_2)$. $sen(w)$ denotes the set of possible senses for word w. $IC(c_1)$ and $IC(c_2)$ are the information content of c_1 and c_2 respectively. $LSuper(c_1, c_2)$ denotes the lowest super-ordinate of c_1 and c_2.

3.3 Latent Semantic Analysis

LSA assumes that there are some underlying structures in the pattern of words. Given training data, LSA first generates a matrix of co-occurrences of each word in each document, and the sequential order of the words is not concerned. Then singular-value decomposition (SVD) is applied. Instead of representing documents and terms directly as vectors of independent words, LSA represents them as continuous values on each of the k orthogonal indexing dimensions derived from the SVD analysis [8]. Thus we harvest a word represented by a vector in a latent semantic space. It is expected that synonyms will be mapped to the same direction in the latent semantic space.

In our experiment, we train LSA in Wikipedia to get semantic vectors. An open source software named S-Space[2] is available to train LSA models. We finally generate words represented by 50-dimensional vectors for experiments.

3.4 Word Embeddings

Deep learning approaches gain focus owing to the great success of their applications in fields like computer visions. Tremendous researches have investigated the effectiveness of deep learning methods on NLP tasks. Word embeddings are trained by deep neural networks, and manage to demonstrate their powers in many traditional NLP tasks. In the field of ontology matching, however, they have never been utilized. Inspired by the potential of word embeddings, they are given great expectations to leverage the semantics of words. We are not intended to describe the training detail because it is complicated and not closely related to ontology matching task itself. But it is necessary to explain the word embeddings we use.

[2] `https://github.com/fozziethebeat/S-Space`

We train our own word embeddings on Wikipedia (version 20130805). *Word2Vec* is an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. The theory is detailed in [18]. In practice, we discard less frequent words that occur less than five times in the whole corpus. We generate unified 50-dimensional word embeddings.

4 Our Matching Algorithm

4.1 Element-level Matching Algorithm

The input of the matching system is two ontologies, i.e., O_1 and O_2. The input ontologies are first parsed by JENA API[3]. Then the entities in both ontologies are preprocessed. We mainly extract the lexical information, e.g., names, labels and comments of an entity, in this step. Then element-level matching is implemented. Finally the evaluation module provides the matching results.

Algorithm 1. Element-level matching algorithm

$alignment, Entity_1, Entity_2 = \phi$
$Entity_1 = C1 \cup P_1 \cup I_1$
$Entity_2 = C2 \cup P_2 \cup I_2$
for each $e_1 \in Entity_1$ **do**
  $maxSim = 0$
  $candidate = null$
  for each $e_2 \in Entity_2$ **do**
    $sim = max\{Sim(name_1, name_2),$
      $Sim(label_1, label_2),$
      $Sim(comment_1, comment_2)\}$
    if $sim > maxSim$ **then**
      $maxSim = sim$
      $candidate = e_2$
    end if
  end for
  $alignment \leftarrow alignment + (e_1, candidate, maxSim)$
end for
return $alignment$;

Our strategy is maximum matching. For every entity e_1 in the source ontology $O_1 = \{C_1, P_1, I_1\}$, find the most similar entity e_2 in the target ontology $O_2 = \{C_2, P_2, I_2\}$. The correspondence (e_1, e_2, con) is then added to the alignment, the confidence con is the similarity between $e1$ and $e2$. The full algorithm is described in Algorithm 1. Notice that in the hybrid method, $Sim(name_1, name_2)$, $Sim(label_1, label_2)$ and $Sim(comment_1, comment_2)$ are the maximal value of edit distance similarity and word embeddings similarity.

[3] `http://jena.apache.org/`

The reason of using maximum matching is that in the element-level matching phase, the main concern is to guarantee recall. The results of the element-level matcher is usually filtered by some constraint and further used by the structure-level matchers. Promoting the recall of element-level matcher is of great importance.

4.2 Sentence Similarity Algorithm

In most scenarios, an entity in ontology is described by sentences consisting of more than one word. Sometimes entities have multi-word names, not to mention their labels and comments. How to get the similarity between these sentences becomes an obstacle for element-level ontology matching. To cope with this problem, we propose a heuristic method to compute sentence similarities. The basic idea of this method is that if more words in one sentence having similar words in the other, the two sentences are more similar. This method is similar to the way of getting the semantic similarity between sentences [15].

Algorithm 2. Sentence Similarity between S_1 and S_2

```
sum = 0
for each w1 ∈ S1 do
    maxSim = 0
    for each w2 ∈ S2 do
        sim = Sim(w1, w2)
        if sim > maxSim then
            maxSim = sim
        end if
    end for
    sum ← sum + maxSim
end for
sum ← sum/N;
return sum;
```

Assume sentence S_1 has N words; sentence S_2 has M words, and $N > M$. The similarity between S_1 and S_2 can be acquired by Algorithm 2. Where $Sim(w_1, w_2)$ is the similarity between word w_1 and word w_2. If w_1 and w_2 are represented by vectors, we use their dot product as similarity. In practice, we found that most words in a sentence have corresponding vectors. If a sentence contains many words that cannot be represented by vectors, it is probably that the sentence cannot be aligned by other sentences. So the proposed algorithm is rational.

In WordNet method, if either w_1 or w_2 is not in WordNet, $Sim(w_1, w_2)$ will be zero. This is common because of the low coverage of WordNet. Likewise, in word embeddings method, if either w_1 or w_2 is not in the training data, $Sim(w_1, w_2)$ will be zero.

5 Experiment

Data Sets: The data sets we use contain two parts:

1) OAEI 2013 benchmark and conference track. The benchmark test suits are systemically generated from a seed bibliographic reference. This seed ontology is modified to generate new ontologies, and the task is matching the new ones to the original ontology. The goal of the conference track is to find alignments within a collection of seven ontologies describing the domain of organizing conferences. The ontologies are from different origins to make sure the heterogeneity.

2) Real-world ontologies, i.e., Freebase, YAGO2 and DBpedia. Freebase has more than 7000 properties, and we chose the most frequent 88 properties. YAGO2 has 125 properties, and we extract 1370 properties from DBpedia.

Evaluation: OAEI 2013 benchmark and conference track both have gold standards respectively, hence the performance could be measured by precision, recall and F-measure. There are dozens of test cases in each track, so the final results are given by the harmonic means according to [21].

$$\begin{cases} H(p) = \frac{\sum_{i=1}^{n} |C_i|}{\sum_{i=1}^{n} |A_i|} \\ H(r) = \frac{\sum_{i=1}^{n} |C_i|}{\sum_{i=1}^{n} |R_i|} \\ H(f) = \frac{2 \times H(p) \times H(r)}{H(p) + H(r)} \end{cases} \tag{4}$$

Where $|A_i|$ denotes the correspondences found by matching system in each alignment, $|C_i|$ refers to correct correspondences and $|R_i|$ is the reference answers.

Real-world ontologies do not have gold standards, so we ask two persons to check the output alignments independently. Only the correspondences accepted by both the checkers are labeled as correct.

Table 1. Results of benchmark

Methods	Precision	Recall	F-measure
Edit Distance	**0.993**	0.622	**0.765**
WordNet (Wup)	0.862	0.510	0.641
WordNet (Lin)	0.990	0.557	0.713
WordNet (Jcn)	0.949	0.535	0.684
LSA	0.993	0.615	0.760
Word Embeddings	0.990	0.616	0.760
Hybrid	0.990	**0.623**	**0.765**

For comparisons we select several methods as follows:

1) Edit distance. Using edit distance to measure the similarities between elements.

2) WordNet. There are three principal ways to compute WordNet similarity. We employ Wu and Palmer [27]'s method to obtain edge-based similarity; employ Lin

[16]'s method to obtain information-based similarity; employ Jiang and Conrath [13]'s method to obtain hybrid WordNet similarity. We use *WordNet (Wup)*, *WordNet (Lin)* and *WordNet (Jcn)* to represent them in the results.

3) LSA. Words are represented by vectors in a latent semantic space after using LSA on training data, thus the similarities can be calculated as cosine similarities. We train LSA on Wikipedia (version 20130805) to acquire semantic vectors.

5.1 Benchmark and Conference

The benchmark results are given in Table 1. Particularly, we conducted experiments on benchmark test case 205, which replaces some names of entities with their synonyms. It is because we believe that word embeddings are adept in dealing with synonyms. The results of test case 205 can be seen in Table 2. The results of conference track are given by Table 3.

Table 2. Results of benchmark test case 205

Methods	Precision	Recall	F-measure
Edit Distance	0.351	0.351	0.351
WordNet (Wup)	0.287	0.258	0.272
WordNet (Lin)	0.386	0.330	0.356
WordNet (Jcn)	0.373	0.320	0.344
LSA	0.358	0.351	0.354
Word Embeddings	0.421	0.412	0.417
Hybrid	**0.433**	**0.433**	**0.433**

Table 3. Results of conference

Methods	Precision	Recall	F-measure
Edit Distance	0.860	0.482	0.618
WordNet (Wup)	0.786	0.469	0.587
WordNet (Lin)	**0.877**	0.466	0.608
WordNet (Jcn)	0.770	0.462	0.578
LSA	0.876	0.462	0.605
Word Embeddings	0.872	0.469	0.610
Hybrid	0.875	**0.482**	**0.622**

In the results, we obtained the following observation:

1) Word embeddings method always achieves higher F-measure than WordNet methods do.

2) Word embeddings mehtod is good at dealing with synonyms.

3) The hybrid method achieves the best performance (in terms of F-measure) in both benchmark and conference track.

5.2 Real-World Ontologies

The matching tasks between these ontologies will yield three alignments. Our algorithm only generates the alignments of properties, because the alignments of classes and individuals are too large to evaluate. The final results are shown in Table 4.

In the results, we obtained the following observation:

Table 4. Results of matching real-world ontologies

Methods	Freebase-DBpedia	Freebase-YAGO2	YAGO2-DBpedia
Edit Distance	30	7	11
WordNet (Wup)	28	3	11
WordNet (Lin)	29	7	12
WordNet (Jcn)	23	6	11
LSA	**33**	7	14
Word Embeddings	**33**	**8**	**15**
Hybrid	**33**	**8**	**15**

1) The found correspondences are relatively scarce, because not all properties have appropriate matching.

2) Word embeddings method finds more correct correspondences than WordNet methods in all the three tasks.

3) The hybrid method does not find more correct correspondences than word embeddings method. It is because edit distance does not contribute to the hybrid method in these three tasks.

5.3 Comparisons among Word Embeddings

In order to discover the influence of using different word embeddings, we utilized two other word embeddings to fulfil the ontology matching tasks. These two word embeddings are *C&W by Turian* [26] and *SENNA* [5]. The comparisons are given by Figure 1 and Figure 2.

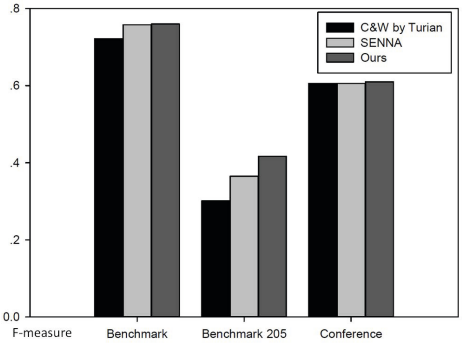

Fig. 1. Comparisons of different word embeddings on OAEI tracks

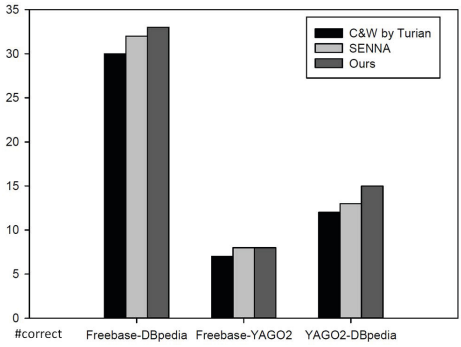

Fig. 2. Comparisons of different word embeddings on real-world ontologies

From the figures we can see that the matching performance is influenced by different word embeddings, and our word embeddings achieve better results than the other two. The reason is that these word embeddings are trained by different data. *C&W by Turian* uses Reuters RCV1 as training data; *SENNA* uses partial Wikipedia and Reuters RCV1 as training data; our training data is the whole Wikipedia. This result indicates that the performance is influenced by the training data.

6 Conclusion

In this paper, we have introduced word embeddings to ontology matching. The experiments show that using word embeddings as a supplement is preferable in ontology matching. The hybrid method combining edit distance and word embeddings achieves the best results.

There are several directions of promotions in the future. The most promising one is training word embeddings specifically. For example, when dealing with biomedical ontologies, we can use relative training data instead of Wikipedia. This amounts to using external resources. It is hopeful since training corpus affect the performance dramatically as our experiments demonstrated. Another possible improvement could be generated by a proper matching strategy. Assigning different weights to semantic similarity and traditions ones appropriately will boost the performance.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (No.61202329, 61272332, 61333018), CCF-Tencent Open Research Fund and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research(ICDD201301).

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)

2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
3. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: Alani, H., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 294–309. Springer, Heidelberg (2013)
4. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. ACM Sigmod Record 35(3), 34–41 (2006)
5. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research 12, 2493–2537 (2011)
6. Cruz, I.F., Antonelli, F.P., Stroe, C.: Agreementmaker: efficient matching for large real-world schemas and ontologies. Proceedings of the VLDB Endowment 2(2), 1586–1589 (2009)
7. Euzenat, J., Shvaiko, P., et al.: Ontology matching. Springer (2007)
8. Foltz, P.W.: Latent semantic analysis for text-based research. Behavior Research Methods, Instruments, & Computers 28(2), 197–202 (1996)
9. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 513–520 (2011)
10. Gruber, T.R., et al.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220 (1993)
11. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence 194, 28–61 (2013)
12. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 235–251 (2009)
13. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint cmp-lg/9709008 (1997)
14. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21(8), 1218–1232 (2009)
15. Li, Y., McLean, D., Bandar, Z.A., O'shea, J.D., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. IEEE Transactions on Knowledge and Data Engineering 18(8), 1138–1150 (2006)
16. Lin, D.: An information-theoretic definition of similarity. In: Proc. 15th International Conf. on Machine Learning, vol. 98, pp. 296–304 (1998)
17. Lin, F., Sandkuhl, K.: A survey of exploiting wordnet in ontology matching. In: Bramer, M. (ed.) Artificial Intelligence in Theory and Practice II. IFIP, vol. 276, pp. 341–350. Springer, Heidelberg (2008)
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS (2013)
19. Mikolov, T., Zweig, G.: Context dependent recurrent neural network language model. In: SLT, pp. 234–239 (2012)
20. Miller, G.A.: Wordnet: a lexical database for English. Communications of the ACM 38(11), 39–41 (1995)
21. Ngo, D., Bellahsene, Z., Todorov, K.: Opening the black box of ontology matching. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 16–30. Springer, Heidelberg (2013)
22. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet:: Similarity: Measuring the relatedness of concepts. Demonstration Papers at HLT-NAACL 2004, pp. 38–41. Association for Computational Linguistics (2004)
23. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering, 158–176 (2013)

24. Socher, R., Lin, C.C., Manning, C., Ng, A.Y.: Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 129–136 (2011)
25. Stoilos, G., Stamou, G., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
26. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 384–394. Association for Computational Linguistics (2010)
27. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138. Association for Computational Linguistics (1994)