Type Vector Representations from Text: An Empirical Analysis

Federico Bianchi, Mauricio Soto, Matteo Palmonari and Vincenzo Cutrona

University of Milan - Bicocca, Viale Sarca 336, Milan, Italy
federico.bianchi,mauricio.soto,palmonari,vincenzo.cutrona@disco.unimib.it

Abstract. Knowledge Graphs (KGs) are abstractions used to represent knowledge in which real-world entities are organized using a type system where types are organized using a sub-type relation: the ontology. A key factor in many applications is to evaluate the similarity between the types of the ontology. Classical measures to evaluate the semantic similarity between types are often based on the structured organization of the sub-type system. In this work, we show that it is possible to use methods coming from Natural Language Processing to embed types in a vector space starting from textual documents. We show that in this representation some of the properties of the hierarchy are still present and that the similarity in this space captures also characteristics that are close to human behavior.

1 Introduction

Knowledge Graphs (KGs) provide abstractions to represent and share knowledge in a structured way that has become popular both in the research community and in the industry. Examples of openly accessible KGs are DBpedia (from which most of following examples are taken) and YAGO, while examples of corporate KGs are the ones developed and used by Google and SpazioDati[1]. Several open models, languages, and technologies have been defined in the context of the semantic Web, like RDF to model data, RDFS and OWL to represent the schema, or, the *ontology*, of the KG. However, some organizations implement the KG abstraction by using other technologies such as, for example, graph databases. We found that three main features are common to different approaches to represent KGs [1]. First, real-world entities, e.g., `dbr:Rome`, are made first class citizens and explicitly represented. Second, relations between these entities, as well as other features of the entities, are represented using several properties, e.g., `dbr:Rome` is located in `dbr:Italy` and has a density of about 2232 per square meter. Third, entities are organized using a rich type system where types are organized using a sub-type relation, e.g., `dbr:Rome` is of type `dbo:City` and `dbo:City` is a sub-type of `dbo:Place`. KGs are often used as a backbone to

[1] https://www.spaziodati.eu

support interoperability between various services and information sources. For example, *entity linking algorithms* find mentions of real-world entities in text, thus supporting different kinds of semantic text analytics that are even marketed by companies (e.g., Dandelion[2] by SpazioDati).

In addition, ontology concepts are organized into sub-type graphs by means of the `rdfs:subClassOf` property. Abstracting away from the specific language used to define the ontology[3], the sub-type graph is a partially ordered set of types[4], which constitutes the backbone of the ontology structure and can be also referred to as *topology*.

The evaluation of semantic similarity between ontology types is an important activity for several tasks like, for example, information retrieval [2].

We can distinguish two types of similarity relationship: semantic similarity and semantic relatedness. Semantic similarity captures the resemblance of entities respect to a more general conceptual term. Instead, semantic relatedness express the existence of a connection between entities independently in which measure they are similar (e.g., gasoline and cars are more related than a car and a bicycle, but these two elements are more similar than the former ones) [3]. Often, semantic similarity is considered as a special case of semantic relatedness. The most popular measures to evaluate the semantic similarity between ontology types are based on the sub-type graph [3, 4].

On the one hand, type similarity measures proposed in the literature tend to consider semantic similarity based on the topology of the type ontology [4]. On the other hand, recent measures of similarity between entities use vector representation of entities derived from the textual corpus [5, 6].

Semantic similarity metrics represent the commonality of two concepts relying on their hierarchical relations [4]. Most of the metrics proposed in the literature are path-based and measure the similarity between two concepts by computing the length of the path between the concepts in the type sub-graph and by considering their hierarchical depth. For example, `dbo:SoccerPlayer` and `dbo:BasketballPlayer` are similar in DBpedia since they share the same parent `dbo:Athlete` in the hierarchy. Known drawbacks of these approaches are that the semantic similarity of any two concepts with the same path length is the same (equal path problem), and also many concepts that share the same depth (hierarchical level) resulting in same similarity (depth problem). The obliviousness of path-based similarities of knowledge coming from concept relations in textual data leads to inconsistencies on the similarity measure. For example, distant nodes in the ontology hierarchy are not necessarily unrelated (e.g., `dbo:SoccerPlayer` and `dbo:SoccerClub`) while siblings types in the ontology

[2] http://dandelion.eu/

[3] In most of this paper we will not adhere to OWL terminology, preferring the terms "types" and "sub-type" to the terms "concept" and "subclass", since the first ones stress the use of concepts that are defined as types of some entities in a KG; however, occasionally, we will use the term "concepts" as equivalent of "types", in particular when referring to related work

[4] The notion of partially ordered set is preferred to the one of taxonomy because more general: many ontologies do not have a tree-shape like taxonomies

might not be equally related (e.g., the similarity between sports vary across the type of sport).

Recent relatedness measures come from Natural Language Processing (NLP) and some of them are based on word embedding techniques. Word embeddings are defined under the distributional hypothesis, which states that words that appear in similar context have similar meaning [7]. Based on this premise word embeddings map words/entities into a small dimension vector space where similarity is based on the co-occurrence of words in a window across the text. Vector assignment attempts to place words that appear in similar contexts closer to each other. Word embeddings also capture intrinsic characteristics of the text like stereotypes [8], thus they are able to extract information about social aspects of the world.

Despite the advantageous features, word embeddings encompass a fundamental drawback: embeddings neglect the type ontological structure and (apparently) fail in representing type hierarchy.

Starting from a recent work [9], in this paper we study a model called *Type to Vector* for measuring the (contextual) similarity between concepts. The model considers text from a set of corpora which are disambiguated into entities and then mapped to (minimal) types. An embedding algorithm is then applied to this type-corpus and similarity between types is defined as the proximity of their vector representation. We show that our method, even without relying on the structure of the ontology, is able to meet some criteria desired by the path-based methods, such that the hierarchical depth assumption (the upper-level concepts in the taxonomy are supposed to be more general and then have a smaller similarity), thus capturing topological properties from the concept ontology.

The paper is organized as follows: Section 2 describes some related works. Section 3 presents the Type to Vector model and Section 4 contains some experimental evidence of the properties of the model. The paper ends outlining some conclusions and future work directions in Section 5.

2 Related Work

In the last few year studies on ontology representations have been conducted. In the case of vector representation, different approaches investigated the usage of embeddings to represent ontologies.

In a recent work [10], a novel instance-based approach is presented. The authors created an ontology in the legal domain and trained a word2vec [11] model with a large corpus of legal documents. The trained model was then used to build the word embedding vectors of the instances and the class labels in the created ontology. Thus, in order to predict the best representative vector for each ontology class, a small number of candidate vectors were calculated using the word embeddings of the instances. The selected candidates are then used to train a machine learning model that predicts the best representative vector for each ontology class. Similarly, another approach [12] makes use of stacked auto-

encoder, to learn the vector representation of each entity from its description bag of words.

A recent approach [13] uses distributional hypothesis based embeddings for ontological representation in which a textual document is generated by considering axioms in an ontology as sentences of a text over which standard method like word2vec can be applied.

Differently from other approaches, here we study a different source of information (i.e., text corpora) that can add details to the representation, but we focus on texts that contain only disambiguated entities in order to train our word2vec model by looking at the concept co-occurrence, without taking into consideration the structure of the ontology, neither its information content (e.g., labels, descriptions, axioms). Approaches to embed entities in the vector space exists [6, 14, 15], but we do not focus on them because they do not directly take care of ontological concepts.

In literature, many path-based metrics for measuring the similarity between concepts are provided. These metrics rely on the ontological topology and consider the length of the path between the concepts and/or their hierarchical depth. One of the first path-based measure (*path*) is based on the shortest path length between concepts [16]. However, relying on the path distance between concepts leads to the *equal path problem*: two concepts with the same path length share the same semantic similarity. Because of this issue, other measures of similarity consider also the depth of concept in the ontology. For example, the *wup* measure computes the concept depth based on the Least Common Subsumer (LCS), which is the first common ancestor of the target concepts [17]. Even if this measure outperforms the previous one, relying on the concept depth has a drawback: concepts at the same hierarchical level share the same similarity (*equal depth problem*). In this scenario, some other path-based approaches start to use external evidence obtained from text in order to overcome both drawbacks.

A recent work on concept similarity proposed the weighted path length (wpath) metric to evaluate the similarity between concepts, by exploiting the statistical Information Content (IC) along with the topology [4]. The IC is computed on text corpora and it is used to assign a higher level to more specific entities. Since IC is based on the concept occurrences in text, each occurrence of a more specific concept implies the occurrence of its ancestor concepts. The objective of this method is to take advantage of structure-based methods for representing the distance between concepts in a taxonomy and to overcome the equal path and depth problems by using the IC between concepts to weight their path length.

3 Type to Vector

In a recent work [9] we have proposed a model to represent both entities and types in the vector space (Typed Entity Embeddings) starting from text. In this work, we want to analyze the properties of the type representation: the Type to Vector model (**T2V**). Starting from a textual corpus we can use Natural

Language Linking techniques to find entities inside text. We can thus create a document that contains only the entities that have been found in text. After this step, we can replace each entity with its own minimal type. Entities can have more than one minimal type: in the current version of the model, we select the first provided by dedicated resources[5]. The generated document contains only types coming from the KG so the last step of the process is to use word embeddings methods like word2vec to generate the embeddings for the types.

Word2vec takes in input a corpus and has two main parameters: the dimensionality of the desired embedded space and a window size that is used to span over the text and to define the co-occurrences context for the words (i.e., context for a word in word2vec is given by the word neighbors based on the window size).

These embeddings capture type-type co-occurrences and thus types that occur in similar context will be close to each other (e.g., `dbo:SoccerPlayer` will appear often near `dbo:SoccerClub`). With this approach we are not using possible relations between types, we are considering only the types of the entities. Figure 1 briefly summarizes the process used to generate the type embeddings. Once the types are embedded it is easy to evaluate the similarity (that is based on the distributional hypothesis) by evaluating the cosine similarity between the vector representations of the types.

Pros and Cons This model provides a fast way to embed an ontology in a vector space in which the distributional hypothesis holds. In the experiments section, we will show that this similarity can capture information that is often not captured by topological measures; this is due to the fact that we are considering an external source of information (i.e., text). Even by considering only the minimal type when building the document with types, we are able to generate vector representation also for types that are not leafs in the hierarchy. This happens because there exist entities that have as minimal type a type that is not a leaf, e.g., `dbo:Agent` and `dbo:Organization`. However, if a type does not appear in the text it is not represented and this can be a limitation of the model. Moreover, the approach depends on the quality of the annotation: if the annotation is wrong the added type will be wrong.

4 Experiments

In this section, we investigate the behavior of our model for representing the ontology in the vector space using distributional semantics. Objectives of these experiments are to show that: 1) linguistic regularities, which are present in classical word embedding models [11], are also present in our model that contains only types of an ontology; 2) some of the information of the hierarchy is preserved in the representation (e.g., more specific types are more similar than less specific types); 3) the similarity measure computed on pairs of types is uncorrelated with topological measures defined in the state of art; 4) the use of word embeddings methods allows to capture and introduce social meaning in the vector space

[5] http://wiki.dbpedia.org/services-resources/documentation/datasets#instancetypes

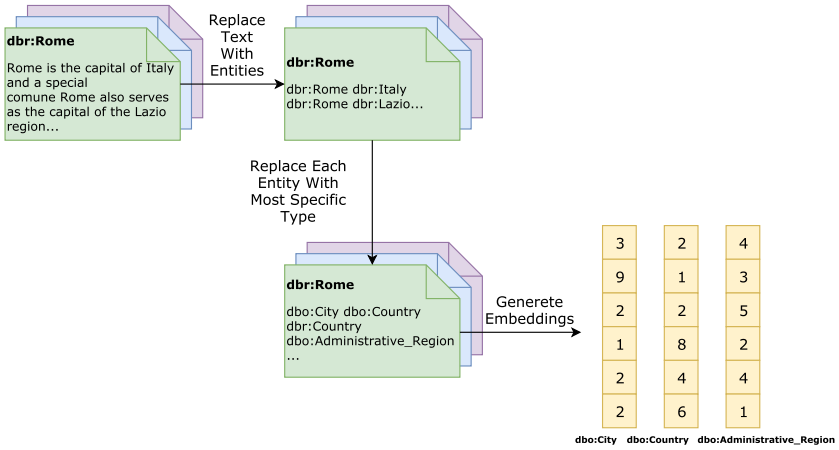

Fig. 1. Process to generate T2V considering text from DBpedia's abstracts

that replicates a human-like behavior in categorization tasks; 5) we can embed multiple classification systems in the same space and evaluate the similarity between them. Our experiments are based on the long abstracts contained in DBpedia 2016-04[6] and DBpedia Spotlight was used as annotator[7]. We used Wikidata 2016-04 dumps[8] for projecting two different categorization systems. The source code of our models and the gold standards are openly available on GitHub[9]. The word2vec implementation that we considered was the skip-gram [11].

4.1 Analogical Reasoning with Types

To verify the quality of the alignment of our embedded representation we consider solving analogies in a similar way to what is usually done to evaluate word embeddings model. Word embeddings are able to represent linguistic regularities by using vector operations like $v(bigger) - v(big) + v(small)$; operations result in a point in the space in which the nearest point should be the correct answer (i.e., $v(smaller)$). We want to apply the same methodology to evaluate our embedding of types. We tested the skip-gram model with two different sizes (100 and 200) with a window of 5.

Dataset Since our representation contains only types, we had to focus on analogies that are made with types. Since we could not find an analogical reasoning gold standard that contained types we created a dataset containing type analogies related to the Sports domain that in DBpedia has a good coverage (e.g.,

[6] http://wiki.dbpedia.org/dbpedia-version-2016-04
[7] http://demo.dbpedia-spotlight.org/
[8] https://tools.wmflabs.org/wikidata-exports/rdf/exports/20160425/
[9] https://github.com/vinid/type2vec

`dbo:SoccerPlayer` is to `dbo:SoccerLeague` as `dbo:BasketballPlayer` is to `dbo:BasketballLeague`). Our dataset contains 868 analogies.
Results Table 1 shows the results obtained by our model on the analogical task in function of the precision (P) (number of analogies solved by considering the closest point to the analogical vector operation of the model) and of the mean relative rank (MRR). Interestingly half of the answers are found in the first position of the dataset and most of them (98%) are found in the top 5 list. One important aspect is that there is a slight variation in the results when considering different parameterizations. Given this result, we decided to use the combination (200, 5) also for the other experimental tasks.

Table 1. Results on the analogical reasoning task

	Precision	**Precision@2**	**Precision@5**	**MRR@2**	**MRR@5**
T2V (200, 5)	**0.50**	**0.85**	**0.98**	**0.68**	**0.77**
T2V (100, 5)	0.47	0.76	0.93	0.61	0.67

4.2 T2V Similarity with respect to depth and distance

As mentioned before, a desired property of our model is that similarity between siblings should increase with the depth of the ontology. This property should holds since the more in-depth we go in an ontology the higher the number of the characteristics shared between nodes: `dbo:SoccerPlayer` and `dbo:BasketballPlayer` should be more similar than their parent (`dbo:Athlete`) and one of the siblings of their parent (e.g., `dbo:Politician`). To evaluate the amount of information that siblings represent we compute a value that measures how much the siblings represent similar things. Given a parent p and the set of its children $C(p)$, we call *Children Information Distribution (CID)* the average similarity of siblings $c_i \in C(p)$. The CID of a parent is thus the average similarity between all the possible pairs of its children.

$$CID(p) = \begin{cases} 1 & \text{if } p \text{ is a leaf or } |C(p)| = 1 \\ \frac{1}{|C(p)|} \sum sim(c_1, c_2)_{\forall c_1, c_2 \in C(p), c_1 \neq c_2}, & \text{otherwise} \end{cases}$$

In Figure 2 we show an actual example of the CID values computed on a small piece of the DBpedia Ontology: the CID of Agent is lower than the one of its children.
Depth Since we believe that siblings that are found in a deeper level of the ontology are more similar to each other we selected all the path from the root (dbo:Thing) to a leaf and we computed the CID of each node on the path (without considering leaves, because they have CID = 1 in the formula) obtaining 54 different paths (409 paths were found but we had to remove leafs from those).

We normalized the depth of each path in [0, 1], thus obtaining the relative depth of each node in the path with its CID. In Figure 3 we plot the relative depth and the CID, which are binned using a window of 0.2 for the relative depth and averaging the CIDs in the bin. The plot shows that the CID increases with

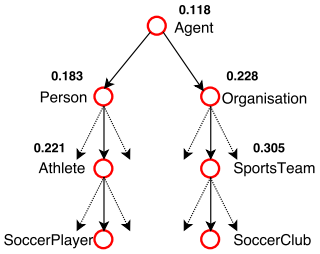

Fig. 2. Nodes that are in deeper sections of the ontology tend to have have an higher CID

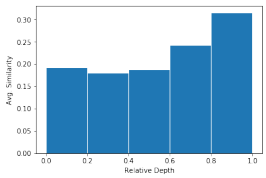

Fig. 3. As we go deeper in the ontology the CID becomes higher

length. We identify an outlier that is the first bar of the plot: it is higher than the second one. This is due to the fact that the type `dbo:Thing` has an higher CID than one of its children, `dbo:Agent`, which is present in most of the paths. Some `dbo:Agent` children are not present in the adopted text corpus (e.g, `dbo:Family`) because they are not used with high frequency. Average CID of the children of Thing is actually 0.36, higher than the CID of Thing itself, 0.18. Interestingly, while topological similarity measures try to force this property [4], in our case this property is intrinsically inherit by the model construction.

Distance We also wanted to compare if in our model there was a relation between path distance and vector similarity of two types. Usually, the semantic similarity should decrease with the increase of the distance. We thus randomly selected pairs of nodes in our representation and computed their distance and their vector similarity. Eventually, we computed the Pearson correlation between these two variables. We found out that there seems to be no relation between topological distance between two nodes and their similarity: correlation is equal to -0.2. This is a result of the closeness of types that are used often together in text (e.g., `dbo:SoccerPlayer` and `dbo:SoccerClub`) but that are far inside the ontology.

4.3 Comparison with Topological Measures

In this section we want to compare the similarity computed by T2V with the similarities computed by state-of-the-art measures. To test the information captured by our measure with respect to the others, we collect all the pairs of types and compute the vector similarity between those pairs using T2V, among with different similarity measures defined in the state of the art. We consider

wpath, *wup* and *path*. Finally, we evaluate the Pearson correlation between these measures.

Results Table 2 shows the results. As we expected, since our measure uses information that is not accessible to topological ones, the correlation is low. Vice versa, our measures can not directly access to the structure of the ontology and thus those measures are more correlated one with each other. Our model tends to give higher similarity scores to those types that are used in the same context (i.e., `dbo:Vein` and `dbo:Artery`).

Table 2. Correlation comparison between T2V and topological measures

	path	**wup**	**wpath**	**T2V**
path	1.00	0.87	0.94	0.30
wup	0.87	1.00	0.93	0.33
wpath	0.94	0.93	1.00	0.36
T2V	0.30	0.33	0.36	1.00

4.4 User Study on Type Similarity

We studied the effect of the similarity between types by considering a simple categorization task in which we involved 5 users that had already some experience with the semantic web.

Methodology We selected 31 nodes from the DBpedia Ontology and for each one we retrieved its most similar sibling and its least similar sibling (which correspond, respectively, to the nearest and to the farthest siblings in the space). As instance, the most similar sibling of `dbo:President` is `dbo:PrimeMinister`, while the least similar is `dbo:Mayor`. Users were given the first node (`dbo:President`) and were asked to decide which of the two siblings they considered more similar. Users were forced to give an answer even in contexts in which it was not immediately clear which element was the most similar (e.g., is `dbo:Skyscraper` more similar to `dbo:Hospital` or `dbo:Museum`?). A strong bias in this experiment is that the two available options were chosen by considering their position in the vector space.

Results Resulting categorizations provided by user were quite similar, the 5 users agreed on many questions. Since the agreement between the user was high we used Gwet AC1 [18] to compute the level of agreement between users, obtaining a level of agreement equal to 0.9, not distant from 1 (that represents unanimity). If we consider the majority vote on the collected answers, we see that the answer is always the most similar sibling. This is an interesting result because it shows that this corpus-based similarity on concepts can capture human-like behavior.

4.5 Projecting Different Classification Systems in the Same Space

In the following, we show that is possible to generalize our model to represent different ontologies in the same vector space. As a result of this representation, we are able to detect similar types of different ontologies in the space. Moreover, if the similarity between two types is high it means that the two are used in the same contexts, and thus they might be representing the same type. This might be useful in the context of equivalent class relations in KG, that allow linking equivalent types of different ontologies.

Methodology We generate a mixed embedding that contains types represented both in DBpedia and Wikidata ontologies by considering DBpedia Ontology and the *instance of* hierarchy provided by Wikidata. The generation of the document is akin to the one presented in Figure 2 with one difference: during the type replacement phase, for each entity we select with probability 0.5 the type coming from the respective Wikidata entity (by first mapping DBpedia's URI to Wikipedia's one, and then to Wikidata's one) or the type of the entity itself in DBpedia. With this approach, we build a mixed corpus that contains types of the two different knowledge bases. Skip-gram is then applied to the corpus to generate embeddings. Differently, from before, the combined representation is embedded in 100-dimensional vector space in which we would like to see similar types from the two different ontology to be close to each other. Our intuition suggests that since we are replacing different types of the same entities, our embedded representation should show equivalent types near to each other (i.e., those types that are used in the same contexts).

Results In Table 3 we show an example of the most similar types. Non-marked pairs are those for which already exists an equivalent class relation in DBpedia. One important aspect of our similarity is that it does not consider syntactic or topological information to find the mappings.

Table 3. Top similar Wikidata entity and DBpedia class pairs

Wikidata (label)	**DBpedia**	**Sim**
Q4498974 (ice hokey team)	HockeyTeam	0.99
Q5107 (continent)	Continent	0.99
Q17374546* (Australian rules football club)	AustralianFootballTeam	0.99
Q3001412* (horse race)	HorseRace	0.98
Q4022 (river)	River	0.98
Q46970 (airline)	Airline	0.98
Q18127 (record label)	RecordLabel	0.98
Q13027888* (baseball team)	BaseballTeam	0.98
Q11424 (film)	Film	0.98
Q1075* (color)	Colour	0.98

Equivalent classes not defined in the KG are also found in the model. Some examples of these types are reported in Table 4.

Table 4. Examples of similar types that have not an equivalent class relation

	Nearest Point	Label	Sim
`dbo:AmericanFootballTeam`	Q17156793	American football team	0.95
`dbo:Earthquake`	Q7944	earthquake	0.91
`dbo:Diocese`	Q3146899	diocese of the Catholic Church	0.93

We also computed the number of correct equivalent classes (by considering DBpedia mappings as a gold standard) we find as top-similar pairs of wikidata and DBpedia types in our representation. Table 5 shows the result of this analysis. It is evident that the number of matched elements decreases as we get far away from the pairs that have the highest similarity. There are some errors that might be considered for a more detailed analysis: `dbo:Aircraft` has Q11436 (aircraft) as equivalent class, while our model suggested Q15056993 (*aircraft family*). This depends on the fact that aircrafts in Wikidata are usually instance of *aircraft family* and subclass of a descendant of *aircraft*. We did not consider the subclass of relation in our embedding and that might be the cause of the error.

Table 5. Percentage of matched equivalent classes for the top similar pairs in the representation

	Top-5	Top-10	Top-15	Top-50	Top-100
Matched%	1.00	0.90	0.80	0.62	0.51

5 Conclusion and Future Work

In this work we have proposed a simple model to represent types of entities in the vector space. We also studied some of the properties of this model and we realized that some of the properties that we had in the ontology (e.g., deeper siblings are more similar to each others than ancestors) are also present in our representation. We also showed with a simple user study that this representation computes similarity in a way that is similar to what human do. Finally we have shown that we can project multiple classification systems in the same vector space with the possibility of having similar types in different ontologies close to each other.

While these results are promising we intend to extend this approach in different way: we would like to combine our findings with topological similarity measures. Also, since we can project multiple representation in the same vector space, we believe that our approach can be used for ontology matching tasks.

References

1. Baoxu Shi and Tim Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems*, 104:123–133, 2016.
2. Angelos Hliaoutakis, Giannis Varelas, Epimenidis Voutsakis, Euripides GM Petrakis, and Evangelos Milios. Information retrieval by semantic similarity. *IJSWIS*, 2(3):55–73, 2006.
3. Philip Resnik et al. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130, 1999.
4. Ganggao Zhu and Carlos A Iglesias. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, 2017.
5. Pierpaolo Basile, Annalina Caputo, Gaetano Rossiello, and Giovanni Semeraro. Learning to rank entity relatedness through embedding-based features. In *NLDB*, pages 471–477. Springer, 2016.
6. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *EMNLP*, volume 14, pages 1591–1601, 2014.
7. Zellig Harris. Distributional structure. In *The Philosophy of Linguistics*. Oxford University Press, New York, 1964.
8. Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.
9. Federico Bianchi and Matteo Palmonari. Joint learning of entity and type embeddings for analogical reasoning with entities. In *NL4AI Workshop, co-located with AI*IA conference*, 2017.
10. V. Jayawardana, D. Lakmal, N. de Silva, A. S. Perera, K. Sugathadasa, and B. Ayesha. Deriving a representative vector for ontology classes with instance word vector embeddings. In *INTECH*, pages 79–84, Aug 2017.
11. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
12. Lirong Qiu, Jia Yu, Qiumei Pu, and Chuncheng Xiang. Knowledge entity learning and representation for ontology matching based on deep neural networks. *Cluster Computing*, 20(2):969–977, Jun 2017.
13. Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *arXiv preprint arXiv:1802.00864*, 2018.
14. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119, 2014.
15. Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.
16. Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE transactions on systems, man, and cybernetics*, 19(1):17–30, 1989.
17. Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of ACL*, pages 133–138. Association for Computational Linguistics, 1994.
18. Kilem Li Gwet. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48, 2008.