ISSN 2007-9737

Order Embeddings for Supervised Hypernymy Detection

Mathias Etcheverry, Dina Wonsever

Universidad de la Repubica, Montevideo,
Grupo de PLN, InCo,
Uruguay

mathiase@fing.edu.uy, wonsever@fing.edu.uy

Abstract. In this work we present a supervised approach to partially order word embeddings, through a learned order embedding, and we apply it in supervised hypernymy detection. We use neural network as an order embedding to map general purpose word embeddings to a partially ordered vector set. The mapping is trained using positive and negative instances of the relationship. We consider two alternatives to deal with compound terms: a character based embedding of an underscored version of the terms, and a convolutional neural network that consumes the word embedding of each term. We show that this distributional approach presents interesting results in comparison to other distributional and path-based approaches. In addition, we observe still good behavior on different sized portions of the training data. This may suggest an interesting generalization capability.

Keywords. Hypernymy, word embedding, order embedding, neural network, Siamese network.

1 Introduction

Hypernymy refers to the general-specific relationship between two lexical terms. Such is the case of biology taxonomies (e.g vertebrate-mammal, mammal-pangolin), professions (e.g. composer-Lennon)[1] and colors (e.g. color-green), among many others. The general term is called the hypernym and the specific one the hyponym. This relationship is crucial in language understanding and generation.

[1]Strictly speaking, Lennon is not a hyponym for composer, it is an instance. In our work we are using a broader notion for the hypernymy relation, including instances of a class as its hyponyms.

In natural language processing, hypernymy can be useful for several tasks such as question answering [7], textual entailment [4] and image detection [18].

A hand-tailored well known resource is WordNet [20]. It is a large lexical database with lexical relations including hypernymy among them. It was originally created for the English language and later other languages have been included through scheme transfer and translations of the English version.

The original English version consumed a considerable human effort for its creation and maintenance. The derived other languages versions suffer from incompleteness and eventually from inadequacies resulting from the transferred scheme. Furthermore, different applications require the expansion of the hypernymy relationship to particular instances like celebrities, song names, video games, and so on. In this context, it is not surprising that automatic hypernymy detection has been an active NLP research area in the last decades.

Supervised hypernymy detection has been mainly addressed based on the use of two sources of corpus based information. On the one hand, the so-called path based approaches, that use the sequence of terms that connects joint occurrences of related pairs (e.g., vertebrates *such as* mammals) [11, 22].

These methods can present low recall due to the difficulty to find both terms in the same context. On the other hand, the distributional approaches use the context information of each term independently.

Several distributional approaches have been carried out through the use of word embeddings [2].

Word embeddings are a popular NLP technique that has been used on various tasks with successful results. It consists on assigning vector representations to words using their contexts in a large text corpus. These methods condense distributional information of words or characters in a large corpus, based on the idea that a word can be characterized by its use [8]. Many methods to learn word embeddings have been proposed recently [19, 23, 17, 13].

HypeNET [26] combines both approaches: distributional and path-based. It is a neural network model that takes as input the word embeddings of two candidate terms and a path based representation. The latter is built from the LSTM [12] representation of the paths distribution in the corpus. They showed the improvement of this approach in comparison to the best distributional one in a dataset created for such purposes. This dataset contains two partitions depending on the intersection of the vocabularies among train, validation and test: a randomly performed and a lexically disjoint split to take into account the possibility of lexical memorization [17].

In this work we propose to build an order embedding as a method for hypernymy detection. An order embedding is a mapping between two partially ordered sets that is injective but not necessarily surjective. Using the order embedding learning strategy presented by Vendrov et al. [29] we realize that is possible to partially order word embeddings and we apply it to perform hypernymy detection. We use, for the order embedding itself, an artificial neural network that consumes pretrained word embeddings and outputs non negative vectors. The network is trained contrastively using positive and negative instances of the relationship.

Compound terms like "grizzly bear" are common in language. To deal with them we consider two alternatives: a character based embedding of an underscored version of the terms, and a representation using the embedding of each word and convolutional layers as the input layers of the neural network. We try different feed forward networks for the mapping. We perform our experiments on a publicly available dataset [26].

We show that this simple approach overcome the results of the best distributional and path-based approaches in same conditions of data usage.

Our use of the order embeddings in conjunction with a neural network exhibits the shape of a Siamese Network [21] with an asymmetric distance measure. Influenced by the use of Siamese Networks in one-shot learning, we study the behavior of the presented model by training it on different slices of the training data.

2 Related Work

Hypernymy detection in NLP can be focused as a supervised or an unsupervised learning task, depending on the available information. Supervised approaches relies on pairs annotated with the information of whether they belong to the relationship or not. On the contrary, unsupervised approaches do not use annotated instances, they rely solely in the distributional inclusion hypothesis [33] or entropy based measures [25].

Supervised approaches have been addressed mainly using two types of information: paths and contexts distributions (or word embeddings). Path-based approaches use the paths of words that connect pairs holding hypernymy relationship. Hand-crafted paths were used as patterns for hypernymy extraction [11]. For example, the path "is a type of" would match cases like "tuna is a type of fish" allowing to conclude that "tuna" is an hyponym of "fish".

In addition, paths in a syntactic dependency tree of joint occurrences in a corpus result useful for hypernymy [27]. In later works, path patterns are generalized using part-of-speech tags and ontology types [22]. The main disadvantage of path-based approaches is that both candidates must occur simultaneously in the same context.

Distributional based approaches use the contexts of the words in a corpus to represent each term. Many methods propose supervised classification after applying a binary vector operator to the pair of term representations. Operators such as vector concatenation [2] and difference were considered [24, 9, 32]. Vylomova et al. studied

vector difference behavior in a wider set of lexical relations and they remarked the importance of negative training data to improve the results [31]. Ustalov et al. performed hypernyms extraction based on projection learning [28]. Instead of classifying the pair of representations, they learned a mapping to project hyponyms embeddings to their respective hypernyms remarking also the importance of negative sampling.

Shwartz et al. combined path-based and distributional information to improve hypernymy detection [26]. They concatenated the embedding of both terms to be classified with a representation of all paths between the terms in a dependency parsed corpus. The representation was built with the average of the LSTM resulting representation of each path. Additionally, they introduced a dataset for lexical entailment where they tested their model.

LEAR (Lexical Entailment Attract-Repel) [30] gives state-of-art performance on hypernymy detection specializing word embeddings based on WordNet constraints. The direction of the asymmetric relation was encoded in the resulting vector norms while cosine distance jointly enforces synonyms semantic similarity. The resulting vectors were specialized simultaneously for lexical relatedness and entailment.

3 Model

In mathematics, an order embedding is a monotone function from one partially ordered set into another. In this work we apply the order embedding presented by Vendrov et al. [29] to hypernymy detection, using for the mapping a neural network that consumes pretrained word embeddings as input. The neural network is trained via back-propagation through supervised examples. We show that the learned transformation can unravel embeddings to detect hypernymy relationship.

The supervised data consist on hypernymy instances (positive examples) and unrelated pairs (negative examples). We consider two alternatives for compound terms: feed forward networks that consume a character based embedding of the whole terms, and convolutional neural networks to obtain the compound term representation through the embedding of each word.

Vendrov et al. already provided an application to hypernymy detection. They showed the representational power of the introduced order embeddings comparing their results to the transitive closure using a randomly split dataset from WordNet constraints. Their application is built using uniquely WordNet constraints. Our work differs in that we pretend to study the capability of pretrained word embedding for hypernymy detection through a supervised trained neural network order embedding.

Particularly, we are interested in the capability of the model to predict the relationship between two terms that have not been seen during training. Note that in their application example the model is not capable to predict an adequate answer for unseen pairs, since it does not have any information of neither of the two terms. In that sense, our work presents more similarities with their application to textual entailment. In the latter they consider a GRU [6] sentence representation using word embeddings as input and performing transfer learning.

We observe that the presented use of order embeddings in conjunction with a neural network exhibits the shape of a Siamese Network [21]. Siamese networks come from the area of computer vision and were introduced in application to signature verification [3]. They mapped image pairs using the same convolutional neural network to take signatures of the same person to equal output vectors. They use vector distance as a measure of equality. Later, Siamese Networks were used in one-shot learning [16] showing strong capabilities to discriminate features of images in scenarios where one image example of each class is observed on training time. In NLP Siamese Networks were considered for sentence similarity using LSTM based representations for input sentences [21].

3.1 Order Embeddings

An order embedding is a function between two partially ordered sets $f : (X, \preceq_X) \to (Y, \preceq_Y)$ that

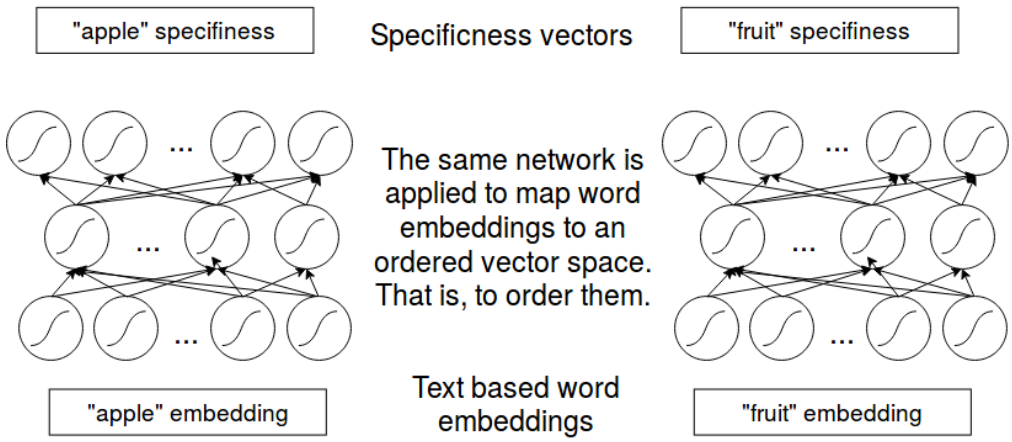

Fig. 1. Siamese Neural Network architecture to map from word to order embeddings

preserves and reflects the order relationships. That is to say, $x_1 \preceq_X x_2$ if and only if $f(x_1) \preceq_Y f(x_2)$.

Note that an order embedding is necessarily an injective function but it may not be surjective, differentiating it from an order isomorphism. In fact, an order embedding provides a way to embed one partially ordered space into another preserving the order structure.

Vendrov et al. present a way to embed an arbitrary a set with an application dependant hierarchical structure into $\Re^m_{\geq 0}$. For that purpose they consider the **reversed product order** on $\Re^m_{\geq 0}$ defined by the conjunction of total order on each coordinate as follows:

$$x \preceq y \iff \bigwedge_{i=1}^{m} x_i \geq y_i, \tag{1}$$

where $x, y \in \Re^m_{\geq 0}$ and x_i and y_i correspond to the i-th component of x and y, respectively. Note that this relationship is anti-symmetric and transitive and that $\vec{0}$ is the top element of the hierarchy.

3.1.1 Loss Function

The partial order relation $(\preceq, \Re^m_{\geq 0})$ defined above allows to define measures to quantify the degree to which a pair of two elements does not satisfy the relationship. Let us consider:

$$E_p(\vec{x}, \vec{y}) = ||max(\vec{0}, \vec{y} - \vec{x})||^2, \tag{2}$$

where $\vec{x}, \vec{y} \in \Re^m_+$ and max is the maximum function component wise. Note that E_p indicates the unrelatedness degree and $E_p(x, y) = 0$ iff $\vec{x} \preceq \vec{y}$.

Additionally, E_p is forced to be higher than a threshold α for unrelated terms through a max-margin loss. Lets consider $\vec{x'}, \vec{y'} \in \Re^m_{\geq 0}$ then:

$$E_n(\vec{x'}, \vec{y'}) = max\{0, \alpha - E_p(\vec{x'}, \vec{y'})\}. \tag{3}$$

Note that $E_n(\vec{x'}, \vec{y'})$ is 0 when $E_p(\vec{x'}, \vec{y'}) \geq \alpha$ guaranteeing that $\vec{x'} \not\preceq \vec{y'}$. Then, summing (2) and (3) the resulting loss function, which consists of minimizing E_p and E_n jointly, has the following expression:

$$L = \sum_{(x,y) \in P} E_p(\vec{x}, \vec{y}) + \sum_{(x',y') \in N} E_n(\vec{x'}, \vec{y'}), \tag{4}$$

where P and N are sets of positive and negative examples, respectively. Note that L is differentiable allowing to fit a mapping to be an order embedding through gradient descent.

3.2 Mapping and Compound Terms

As we commented before, we consider neural networks as order embeddings to embed word embedding, partially ordered by hypernym relation, into a non negative vector space ordered by the reversed product order.

We consider equation (4) for the loss function to train the neural network. We expect that if word embeddings contains the needed information to distinguish hypernymy it could be revealed by the learned transformation.

So, lets consider $f_W : \Re^n \to \Re^m_{\geq 0}$ to be a neural network with weights W then, using (4), the loss function is:

$$\sum_{(x,y)\in P} E_p(f_W(x), f_W(y)) + \sum_{(x',y')\in N} E_n(f_W(x'), f_W(y')). \tag{5}$$

The used dataset contains compounds terms, that is to say, terms constituted by many words (e.g. "contemporary art"). We considered two variants for compound terms treatment: 1) One-dimension convolutional layers as input layers to represent the compound term from the word embedding of each of its parts. 2) A FastText generated representation replacing spaces by underscores. This representation is then the input of a feed forward network.

We performed most of our experiments considering the second approach since it presents better results and allows to use the complete dataset because absence of out of vocabulary words. Figure 1 shows a diagram of the model.

The activity function of the network defines the output values. Hence, for this model we were limited to consider activity functions with non negative outputs (such as sigmoid function and ReLU). We considered three variants of feed forward networks according to their activity functions: (1) ReLU network, (2) SELU-ReLU a network with a ReLU output layer and SELU [15] functions on its hidden layers, and (3) tanh-sigmoid a network that used sigmoid function on its output layer and tanh on its hidden layers.

4 Experiments

In this section we describe the experiments conducted, the used resources and the model parameters. First we describe the supervised dataset and word embeddings used, followed by the model structure and parameters. We conclude this section with comments about the results and an error analysis.

4.1 Datasets and Word Embeddings

We use the dataset introduced by Shwartz et al. [26] to perform our experiments. Such dataset is constituted by related and unrelated pairs of words. It was created using distant supervision from a variety of knowledge resources such as WordNet [20] and DBPedia [1], among others.

The dataset provides two variants: lexical and random split. Each variant consists of a division into train, validation and test sets. The concept of lexical split refers to a partitioning without lexical intersection between any of the three parts. That is, if a pair occurs in one subset it will not occur in any pair of the other two subsets. The other split was performed randomly. The dataset size information is detailed on Table 1.

For the word embeddings we consider the publicly available English GloVe 6B[2] and the FastText [13] vectors trained on English Wikipedia with default parameters[3].

The supervised data presented out - of - vocabulary (oov) terms in GloVe 6B embeddings and we discarded the affected tuples for training and evaluation. The amount of discarded examples varies between 45% and 50% on each partition.

In the case of the embeddings built using FastText there is not a problem of out-of-vocabulary terms, as they are character based.

[2] http://nlp.stanford.edu/data/glove.6B.zip
[3] http://mattmahoney.net/dc/enwik9.zip

Table 1. Shwartz's dataset size for random and lexical splits

		Positive	Negative	Total
Random	Train	9,942	39,533	49,475
	Valid Random	681	2,853	3,534
	Test Rand.	3,512	14,158	17,670
Lexical	Train	4,067	16,268	20,335
	Valid	270	10,80	1,350
	Test	1,322	5,288	6,610

Table 2. Results on test set

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Best Distributional [26]	0.901	0.637	0.746	0.754	0.551	0.637
HypeNET Integrated [26]	0.913	**0.890**	0.901	0.809	0.617	0.700
Siamese ReLU	0.936	0.876	**0.905**	**0.958**	0.615	0.749
Siamese SELU-ReLU	0.932	0.845	0.887	0.740	**0.872**	**0.801**
Siamese tanh-sigmoid	**0.967**	0.836	0.897	0.788	0.756	0.771

4.2 Mapping Details

We tried several hyperparameters configurations for the neural networks. We considered ReLU, tanh, sigmoid and SELU layers. For the firsts three with use dropout between them and for the latter we use alpha dropout. We consider a 0.2 of drop probability for dropout and 0.1 for alpha dropout. We initialize SELUs with LeCun normal initialization and ReLUs using Glorot uniform initializer [10].

We observed improved behavior on the model with SELUs for the firsts layers and ReLU units in the output layer. For the convolutional approach, we considered one and two convolutional layers next to the input, with a convolution size of two words vectors, 64 filters each, and max pooling. The convolutional approach did not improves the results obtained by the underscored FastText approach. We think that compound terms are not so numerous and varied to take advantage of the convolutions.

We train our models using Adam [14] with a learning rate of 5×10^{-4} over mini-batches of 64 examples. We stopped the training using early stopping and we checked out the best model in the whole run against the validation set.

We considered the model output classified as positive if E_p is lesser than 0.02 and we use 1.0 as margin for negative examples margin. We used Keras [5] to implement our models.

4.3 Results

In Figure 2 we show the SELU-ReLU model accuracy evolution in the lexical split on the training and validation sets. It can be seen the joint progress of accuracy on train and validation sets, suggesting the capability of the model to distinguish hypernymy relation between terms that have not seen during training.

We evaluate our models using precision, recall and F measures. We present the obtained results on Table 2. We include for comparison the results of the best distributional model reported by Shwartz et al. [26] and HypeNET combined. The reported results are the best, according to the validation set, of three runs of a three layered network of 600, 400 and 200 units on input, hidden and output layer, respectively.

4.4 Results on Reduced Data

Inspired by the success of Siamese Networks in one shot learning we study the performance of the considered model restricting the available training data. In Figure 3 we show the F measure obtained with the SELU-ReLU model trained on gradually increased sizes of the available training data. Note that the results increase rapidly in the first 20% of the training data in both, random and lexical, splits.

We include the results detailing precision and recall in Table 3. Note that the model first achieves high coverage and then seems to refine its results.

4.5 Results Analysis

In this section we include some analysis of the results obtained with the SELU-ReLU model on the lexical split dataset.

In Table 4 we include the confusion matrix. Note that 169 pairs were hypernyms that the model was unable to detect. Of those 169, 100 present a value lesser that 1.0 for the asymmetric score. We compare against 1.0 because that is the chosen margin for negative examples.

A sampling of pairs where the model fails to predict correctly can be found on Table 5. We note that many of the positive pairs that the model fail to predict correspond to ambiguous terms. In the samples presented, note for example that *stubbs* refers to the surname of William Stubbs but it can be confused with the artist George Stubbs, among others. The same stands for **sting**, that can be confused with the singer.

Regarding to the pairs that the model wrongly detects as related we detect that the term *novel* is involved in 129 instances of the 405 false positives. We detect that many of this terms are incorrectly labelled in the dataset. For example, terms as *pollyanna*, *aelita* and *jaws*, are presented as negative examples when in fact they are positive hyponyms of *novel*. Although, *novel* is unique massive hypernym that fails that have been detected, there are other examples of terms that were predicted as negative and seems to be wrongly labelled in the dataset.

In Table 6 we show a sampling pairs that seems to be incorrectly labelled as negative and the model predict as positive. Note that this examples principally correspond to occupations (like writers or actors) and creations (like novels). However, there are also other pairs that seem to

Table 3. SELU-ReLU results on different training sizes

	P_{rand}	R_{rand}	P_{lex}	R_{lex}
5%	0.703	0.899	0.495	0.893
10%	0.694	0.904	0.495	0.895
20%	0.712	0.916	0.574	0.896
30%	0.909	0.869	0.583	0.862
50%	0.916	0.866	0.611	0.912
80%	0.925	0.862	0.630	0.889

Table 4. SELU-ReLU model confussion matrix in lexical split

	False	True
False	4883	405
True	169	1153

Table 5. SELU-ReLU false negative samples

Hyponym	Hypernym
building	structure
contentment	happiness
diver	swimmer
cosmos	flower
moment	present
stubbs	historian
sting	pain

Table 6. SELU-ReLU false positve samples that seems incorrectly labelled

Hyponym	Hypernym
voltaire	writer
bill mantlo	writer
fledgling	novel
summerland	novel
sathyaraj	actor
ferdinand de saussure	linguist
menecrates	sculptor
nicomedes	mathematician
kofax	software
encarta	encyclopedia
abode	place
beretta	weapon

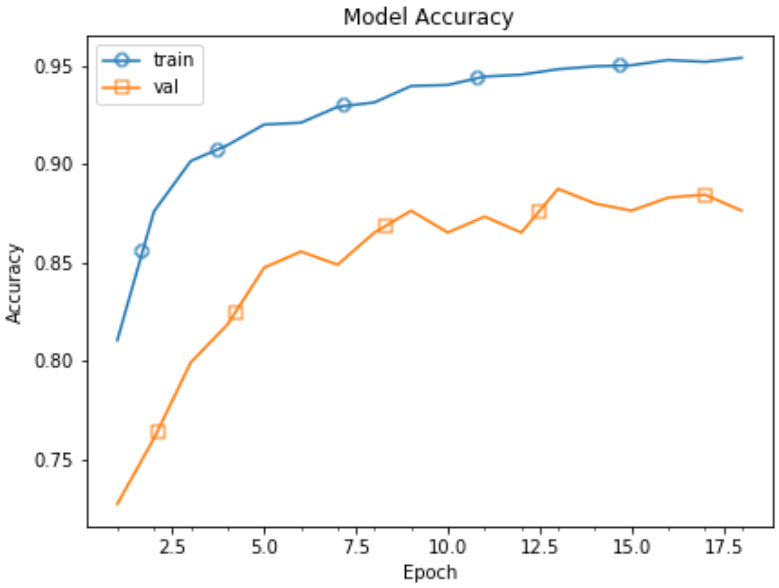

Fig. 2. Accuracy evolution on each epoch in the SELU-ReLU model

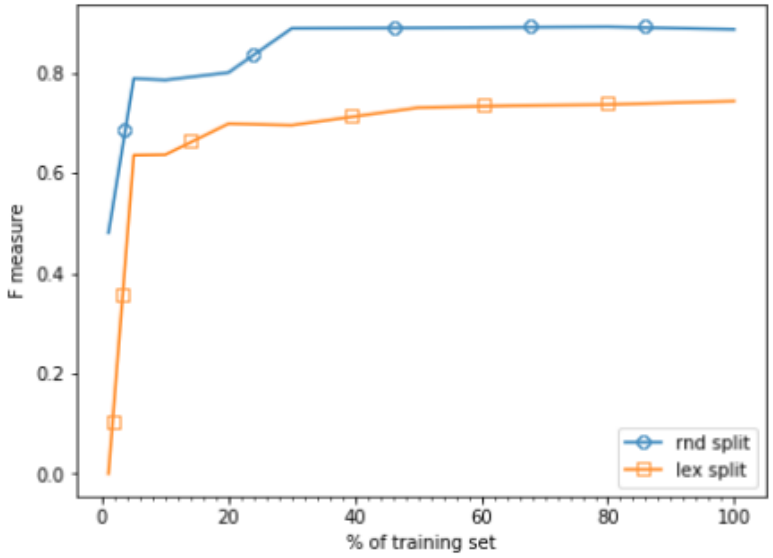

Fig. 3. F measure result of SELU-ReLU model on different portions training set.

be incorrectly labelled that does not correspond to particular instances such as occupations or creations.

For example, terms like *abode* as hyponym of *place* and *beretta* as hyponym of *weapon* are predicted as related while in the data appear as negative examples.

Finally, the Table 7 presents a sample of terms that have been incorrectly predicted as related. The difference between this samples and the presented in Table 6 is that these ones does not

Table 7. SELU-ReLU false positve samples

Hyponym	Hypernym
stump	tree
rice	sake
tofee	sugar
nucleus	brain
wjre	country
edge	limb

seem to be incorrectly labelled. However, note that in the case of *stump* as hyponym of *tree*, although it is not a variety of tree, it is a tree that have fallen or that have been cut down. And, in the case of *rice* and *sake* there is and composed-of relation, since the sake is made of rice.

5 Conclusion

We present a distributional model for supervised hypernym detection. The model relies on learning an order embedding from word embeddings into a non negative vector space. For the order embedding itself we consider feed forward artificial neural networks and we explore different model configurations.

We show that this approach gives competitive results on a publicly available dataset in comparison to the best distributional and path-based approaches reported on same data. We study the performance of the model restricting the available training data. We found that the model give relative good results using less than 10% of the training data. This suggest that the model tends to learn the order from data even when relatively few examples are provided.

References

1. **Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007).** Dbpedia: A nucleus for a web of open data. *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, Springer-Verlag, Berlin, Heidelberg, pp. 722–735.
2. **Baroni, M., Bernardi, R., Do, N., & Shan, C. (2012).** Entailment above the word level in distributional semantics. *EACL*, The Association for Computer Linguistics, pp. 23–32.
3. **Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993).** Signature verification using a "Siamese" time delay neural network. *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 737–744.
4. **Chen, Q., Zhu, X., Ling, Z., Inkpen, D., & Wei, S. (2017).** Natural language inference with external knowledge. *CoRR*, Vol. abs/1711.04289.
5. **Chollet, F. et al. (2015).** Keras. `https://keras.io`.
6. **Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014).** Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, Vol. abs/1412.3555.
7. **Clark, P., Fellbaum, C., & Hobbs, J. (2007).** Using and extending wordnet to support question-answering. *GWC 2008: 4th Global WordNet Conference, Proceedings*.
8. **Firth, J. (1957).** *Papers in linguistics, 1934-1951*. Oxford University Press.
9. **Fu, R., Guo, J., Qin, B., Che, W., Wang, H., & Liu, T. (2014).** Learning semantic hierarchies via word embeddings. *ACL (1)*, The Association for Computer Linguistics, pp. 1199–1209.
10. **Glorot, X. & Bengio, Y. (2010).** Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, volume 9 of *JMLR Proceedings*, JMLR.org, pp. 249–256.
11. **Hearst, M. A. (1992).** Automatic acquisition of hyponyms from large text corpora. *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pp. 539–545.
12. **Hochreiter, S. & Schmidhuber, J. (1997).** Long short-term memory. *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780.
13. **Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016).** Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.
14. **Kingma, D. P. & Ba, J. (2014).** Adam: A method for stochastic optimization. *CoRR*, Vol. abs/1412.6980.

15. **Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017).** Self-normalizing neural networks. In **Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R.**, editors, *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 971–980.
16. **Koch, G., Zemel, R., & Salakhutdinov, R. (2015).** Siamese neural networks for one-shot image recognition. *32 nd International Conference on Machine Learning*, pp. 1–8.
17. **Levy, O., Remus, S., Biemann, C., & Dagan, I. (2015).** Do supervised distributional methods really learn lexical inference relations? **Mihalcea, R., Chai, J. Y., & Sarkar, A.**, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, The Association for Computational Linguistics, pp. 970–976.
18. **Marszalek, M. & Schmid, C. (2007).** Semantic hierarchies for visual object recognition. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference*, pp. 1–7.
19. **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** Efficient estimation of word representations in vector space. *CoRR*, Vol. abs/1301.3781.
20. **Miller, G. A. (1995).** Wordnet: A lexical database for english. *Commun. ACM*, Vol. 38, No. 11, pp. 39–41.
21. **Mueller, J. & Thyagarajan, A. (2016).** Siamese recurrent architectures for learning sentence similarity. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, AAAI Press, pp. 2786–2792.
22. **Nakashole, N., Weikum, G., & Suchanek, F. M. (2012).** PATTY: A taxonomy of relational patterns with semantic types. *EMNLP-CoNLL*, ACL, pp. 1135–1145.
23. **Pennington, J., Socher, R., & Manning, C. D. (2014).** Glove: Global vectors for word representation. **Moschitti, A., Pang, B., & Daelemans, W.**, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, ACL, pp. 1532–1543.
24. **Roller, S., Erk, K., & Boleda, G. (2014).** Inclusive yet selective: Supervised distributional hypernymy detection. *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, Dublin, Ireland, pp. 1025–1036.
25. **Santus, E., Lenci, A., Lu, Q., & Schulte im Walde, S. (2014).** Chasing hypernyms in vector spaces with entropy. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pp. 38–42.
26. **Shwartz, V., Goldberg, Y., & Dagan, I. (2016).** Improving hypernymy detection with an integrated path-based and distributional method. *CoRR*, Vol. abs/1603.06076.
27. **Snow, R., Jurafsky, D., & Ng, A. Y. (2004).** Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pp. 1297–1304.
28. **Ustalov, D., Arefyev, N., Biemann, C., & Panchenko, A. (2017).** Negative sampling improves hypernymy extraction based on projection learning. *EACL (2)*, Association for Computational Linguistics, pp. 543–550.
29. **Vendrov, I., Kiros, R., Fidler, S., & Urtasun, R. (2015).** Order-embeddings of images and language. *CoRR*, Vol. abs/1511.06361.
30. **Vulic, I. & Mrksic, N. (2017).** Specialising word vectors for lexical entailment. *CoRR*, Vol. abs/1710.06371.
31. **Vylomova, E., Rimell, L., Cohn, T., & Baldwin, T. (2016).** Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. *ACL (1)*, The Association for Computer Linguistics, pp. 1671–1682.
32. **Weeds, J., Clarke, D., Reffin, J., Weir, D. J., & Keller, B. (2014).** Learning to distinguish hypernyms and co-hyponyms. *COLING*, ACL, pp. 2249–2259.
33. **Zhitomirsky-Geffet, M. & Dagan, I. (2005).** The distributional inclusion hypotheses and lexical entailment. *ACL*, pp. 107–114.

Article received on 30/10/2019; accepted on 11/03/2020.
Corresponding author is Mathias Etcheverry.