Putting RDF2vec in Order

Jan Portisch[1,2][0000−0001−5420−0663] and Heiko Paulheim[1][0000−0003−4386−8195]

[1] Data and Web Science Group, University of Mannheim, Germany
{jan, heiko}@informatik.uni-mannheim.de
[2] SAP SE Business Technology Platform — One Domain Model, Walldorf, Germany
jan.portisch@sap.com

Abstract. The RDF2vec method for creating node embeddings on knowledge graphs is based on word2vec, which, in turn, is agnostic towards the position of context words. In this paper, we argue that this might be a shortcoming when training RDF2vec, and show that using a word2vec variant which respects order yields considerable performance gains especially on tasks where entities of different classes are involved.[3]

Poster Submission

Keywords: RDF2vec · knowledge graphs · knowledge graph embeddings · machine learning

1 Introduction

RDF2vec [13] is a representation learning approach for entities in a knowledge graph. The basic idea is to first create *sequences* from a knowledge graph by starting random walks from each node. These sequences are then fed into the *word2vec* algorithm [7] for creating word embeddings, with each entity or property in the graph being treated as a "word". As a result, a fixed-size feature vector is obtained for each entity.

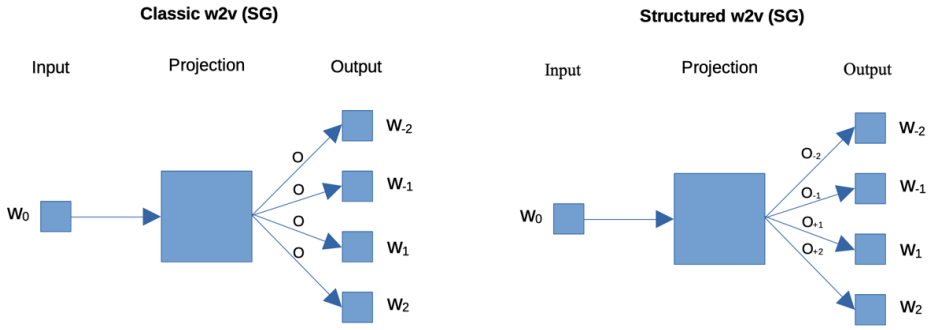

Fig. 1. Classic word2vec vs. Structured word2vec

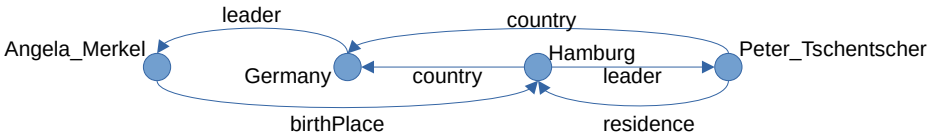

Fig. 2. Example knowledge graph

Word2vec is a well-known neural language model to train latent representations (i.e., fixed size vectors) of words based on a text corpus. Its objective is either to predict a word w given its context words (known as continous bag of words or CBOW), or vice versa (known as skip gram or SG).

Given the context k of a word w, where k is a set of preceding and succeeding words of w, the learning objective of word2vec is to predict w. This is known as *continuous bag of words* model (CBOW). The *skip-gram* (SG) model is trained the other way around: Given w, k has to be predicted. Within this training process, the size of k and is also known as *window* or *window size*.

One shortfall of the original original word2vec approach is its insensitivity to the relative positions of words. It is, for instance, irrelevant whether a word is preceding or succeeding w, and the actual distance to w is not considered. This property of word2vec is ideal to cope with the fact that in many languages, the same sentence can be expressed with different word orderings (cf. *Yesterday morning, Tom ate bread* vs. *Tom ate bread yesterday morning*). In contrast, walks extracted from knowledge graphs, the semantics of the underlying nodes differ depending on the position of an entity in the walk, as the following examples illustrates.

Fig. 2 depicts a small excerpt of a knowledge graph. Among others, the following walks could be extracted from the graph:

```
Hamburg -> country -> Germany            -> leader     -> Angela_Merkel
Germany -> leader  -> Angela_Merkel      -> birthPlace -> Hamburg
Hamburg -> leader  -> Peter_Tschentscher -> residence  -> Hamburg
```

If an RDF2vec model is trained for the entities in the center (i.e., `Germany`, `Angela_Merkel`, and `Peter_Tschentscher`), all of the sequences share exactly two entities in their context (`Hamburg` and `leader`), i.e., they will be projected equally close in the vector space. However, a model respecting positions would particularly differentiate the different meanings of `leader` (i.e., whether someone/thing *has* or *is* a leader), and the different *roles* of involved entities (i.e., `Hamburg` as a place of birth or a residence of a person, or being located in a country). Therefore, it would map the two politicians closer to each other than to `Germany`.

Ling et al. [6] present an extension to the word2vec algorithm, known as *structured word2vec*, which incorporates the positional information of words. This is achieved by using multiple encoders (CBOW) respectively decoders (SG) depending on the position of the context words. An illustration for SG can be found in Figure 1 where it is visible that the classic component uses only one

output matrix O which maps the embeddings to the output while the structured approach uses one output matrix per position in the window (e.g. O_{+1} for the subsequent word to w_0).

In this paper, we present $RDF2vec_{oa}$, an *order aware* variant of RDF2vec obtained by changing the training component from word2vec to structured word2vec, and show promising preliminary results.

2 Related Work

RDF2vec was one of the first approaches to adopt statistical language modeling techniques to knowledge graphs. Similar approaches, such as *node2vec* [4] and *DeepWalk* [11], were proposed for unlabeled graphs while knowledge graphs are labeled by nature, i.e., they contain different types of edges.

Other language modeling techniques that have been adapted for knowledge graphs include GloVe [9], which yielded *KGlove* [2], and BERT [3], which yielded *KG-BERT* [16].

Variants of RDF2vec include the use of different heuristics for biasing the walks [1]; [15] evaluate multiple heuristics for biasing the walks or alternative walk strategies. Very few authors tried to change the training objective of RDF2vec. Besides word2vec, the GloVe [10] algorithm has also been used [2].

3 Experiments and Preliminary Results

We use jRDF2vec[4] [12] to generate random walks and Ling et al.'s structured word2vec implementation[5] to train an embedding based on the walks.

For the embeddings, we use the DBpedia 2016-04 dataset. We generated 500 random walks for each node in the graph with a depth of 4 (node hops). word2vec and structured word2vec were trained using the same set of walks and the same training parameters: *SG*, $window = 5$, and $size \in \{100, 200\}$.

We evaluate both, the classic and the position aware RDF2vec approach, on a variety of different tasks and datasets. For our evaluation, we use the *GEval* framework [8]. We follow the setup proposed in [14] and [8]. Those works use data mining tasks with an external ground truth. Different feature extraction methods – which includes the generation of embedding vectors – can then be compared using a fixed set of learning methods. Overall, we evaluate our new embedding approach on six tasks using 20 datasets altogether. The evaluation is conducted on six different downstream tasks – classification and regression, clustering, determining semantic analogies, and computing entity relatedness and document similarity, the latter based on entities mentioned in the documents.

The results are presented in Table 1. When comparing the classic to the order aware embeddings, it is visible that the performances are very similar on most tasks such as classification. A first observation is that we cannot observe

[4] `https://github.com/dwslab/jRDF2Vec`

[5] `https://github.com/wlin12/wang2vec`

Table 1. Results of RDF2vec$_{classic}$ (c-100, c-200) and RDF2vec$_{oa}$ (oa-100, oa-200) trained with 100 and 200 dimensions respectively. The best value in each dimension group is printed in bold, the overall best value is additionally underlined.

Task	Metric	Dataset	c-100	oa-100	c-200	oa-200
Classification	ACC	AAUP	**0.693**	0.679	**0.692**	0.683
	ACC	Cities	**0.793**	**0.793**	0.798	**0.807**
	ACC	Forbes	**0.629**	0.607	**0.635**	0.630
	ACC	Metacritic Albums	0.783	**0.799**	0.788	**0.792**
	ACC	Metacritic Movies	**0.757**	0.736	**0.763**	0.748
Clustering	ACC	Cities/Countries (2k)	0.755	**0.939**	0.758	**0.946**
	ACC	Cities/Countries	**0.786**	0.785	0.7624	**0.766**
	ACC	Cities/Albums/Movies /AAUP/Forbes	**0.932**	0.931	0.861	**0.929**
	ACC	Teams	0.969	**0.971**	0.892	**0.945**
Regression	RMSE	AAUP	65.151	**62.624**	66.301	**65.077**
	RMSE	Cities	12.726	**11.220**	14.855	**13.484**
	RMSE	Forbes	**34.290**	34.340	36.460	**35.967**
	RMSE	Metacritic Albums	11.366	**11.215**	**11.528**	11.651
	RMSE	Metacritic Movies	**19.091**	19.530	**19.078**	19.432
Semantic Analogies	ACC	Capital-Countries	0.852	**0.990**	0.872	**0.949**
	ACC	Capital-Countries (all)	0.832	**0.933**	**0.901**	0.896
	ACC	Currency-Country	0.417	**0.520**	**0.537**	0.441
	ACC	City-State	0.5577	**0.607**	0.555	**0.627**
Entity Relatedness	Harmonic Mean	-	**0.726**	0.716	**0.747**	**0.747**
Document Similarity	Kendall Tau	-	**0.405**	0.373	**0.350**	0.325

significant performance drops on any of the tasks when switching from classic to order aware RDF2vec embeddings. However, significant performance increases can be observed on clustering tasks and on semantic analogy tasks, which are the tasks where entities of different classes are involved (whereas the classification and regression tasks deal with entities of the same class, e.g., cities or countries). The order aware RDF2vec configuration with 100 dimensions achieved on 7 datasets the overall best results and outperforms its classic configuration with the same dimension on 10 datasets partly with significantly better outcomes. On the other hand, in most cases where the classic variant performs better, it does so by a smaller margin. Thus, in general, the order-aware variant can be used safely without performance drops, and in some cases with significant performance gains.

4 Summary and Future Work

In this paper, we presented a position aware variant of RDF2vec together with first very promising evaluation results. In the future, we plan to conduct more

thorough analyses, analyzing which knowledge graph characteristics and downstream tasks benefit most from the ordered variant, and which do not. For example, we believe that graphs with a small set of predicates, or graphs which have all symmetric, inverse, and transitive relations materialized [5], can benefit more from using the ordered variant.

Furthermore, we plan to analyze how the ordered variant can be integrated into other RDF2vec configurations and flavours, such as different biased walks [2], or RDF2vec Light [12].

References

1. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Biased graph walks for RDF graph embeddings. In: WIMS 2017. pp. 21:1–21:12. ACM (2017)
2. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Global RDF vector space embeddings. In: ISWC 2017. LNCS, vol. 10587, pp. 190–207. Springer (2017)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: ACM SIGKDD 2016. pp. 855–864 (2016)
5. Iana, A., Paulheim, H.: More is not always better: The negative impact of a-box materialization on rdf2vec knowledge graph embeddings. In: Proceedings of the CIKM 2020 Workshops (2020)
6. Ling, W., Dyer, C., Black, A.W., Trancoso, I.: Two/too simple adaptations of word2vec for syntax problems. In: NAACL HLT 2015. pp. 1299–1304. ACL (2015)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
8. Pellegrino, M.A., Altabba, A., Garofalo, M., Ristoski, P., Cochez, M.: Geval: A modular and extensible evaluation framework for graph embedding techniques. In: ESWC (2020)
9. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP 2014. pp. 1532–1543 (2014)
10. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP 2014. pp. 1532–1543. ACL (2014)
11. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: ACM SIGKDD 2014. pp. 701–710 (2014)
12. Portisch, J., Hladik, M., Paulheim, H.: Rdf2vec light - A lightweight approachfor knowledge graph embeddings. In: ISWC Posters and Demos (2020)
13. Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: Rdf2vec: RDF graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019)
14. Ristoski, P., de Vries, G.K.D., Paulheim, H.: A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In: ISWC (2016)
15. Vandewiele, G., Steenwinckel, B., Bonte, P., Weyns, M., Paulheim, H., Ristoski, P., Turck, F.D., Ongenae, F.: Walk extraction strategies for node embeddings with rdf2vec in knowledge graphs. CoRR **abs/2009.04404** (2020)
16. Yao, L., Mao, C., Luo, Y.: Kg-bert: Bert for knowledge graph completion. arXiv preprint arXiv:1909.03193 (2019)