Making Fast Graph-based Algorithms with Graph Metric Embeddings

Andrey Kutuzov[†], **Mohammad Dorgham**[‡], **Oleksiy Oliynyk**[‡], **Chris Biemann**[‡], and **Alexander Panchenko**[⋆,‡]

[†]Language Technology Group, University of Oslo, Oslo, Norway
[‡]Language Technology Group, Universität Hamburg, Hamburg, Germany
[⋆]Skolkovo Institute of Science and Technology, Moscow, Russia

Abstract

The computation of distance measures between nodes in graphs is inefficient and does not scale to large graphs. We explore dense vector representations as an effective way to approximate the same information: we introduce a simple yet efficient and effective approach for learning graph embeddings. Instead of directly operating on the graph structure, our method takes structural measures of pairwise node similarities into account and learns dense node representations reflecting user-defined graph distance measures, such as e.g. the shortest path distance or distance measures that take information beyond the graph structure into account. We demonstrate a speed-up of several orders of magnitude when predicting word similarity by vector operations on our embeddings as opposed to directly computing the respective path-based measures, while outperforming various other graph embeddings on semantic similarity and word sense disambiguation tasks and show evaluations on the WordNet graph and two knowledge base graphs.

When operating on large graphs, such as transportation networks, social networks, or lexical resources, the need for estimating similarities between nodes arises. For many domain-specific applications, custom graph node similarity measures $sim : V \times V \rightarrow \mathbb{R}$ have been defined on pairs of nodes V of a graph $G = (V, E)$. Examples include travel time, communities, or semantic distances for knowledge-based word sense disambiguation on WordNet (Miller, 1995). For instance, the similarity s_{ij} between the cup.n.01 and mug.n.01 synsets in the WordNet is $\frac{1}{4}$ according to the inverted shortest path distance as these two nodes are connected by the undirected path cup → container ← vessel ← drinking_vessel ← mug.

In recent years, a large variety of such node similarity measures have been described, many of which are based on the notion of a random walk (Fouss et al., 2007; Pilehvar and Navigli, 2015; Lebichot et al., 2018). As given by the structure of the problem, most such measures are defined as traversals of edges E of the graph, which makes their computation prohibitively inefficient.

To this end, we propose the *path2vec* model[1], which solves this problem by decoupling development and use of graph-based measures, and – in contrast to purely walk-based embeddings – is trainable to reflect custom node similarity measures. We represent nodes in a graph with dense embeddings that are good in approximating such custom, e.g. application-specific, pairwise node similarity measures. Similarity computations in a vector space are several orders of magnitude faster than computations directly operating on the graph.

First, effectiveness of our model is shown *intrinsically* by learning metric embeddings for three types of graphs (WordNet, FreeBase, and DBPedia), based on several similarity measures. Second, in an *extrinsic* evaluation on the Word Sense Disambiguation (WSD) task (Navigli, 2009) we replace several original measures with their vectorized counterparts in a known graph-based WSD algorithm by Sinha and Mihalcea (2007), reaching comparable levels of performance with the graph-based algorithms while maintaining computational gains.

The main contribution of this paper is the demonstration of the effectiveness and efficiency of the *path2vec* node embedding method (Kutuzov et al., 2019). This method learns dense vector embeddings of nodes V based on a user-defined custom similarity measure sim, e.g. the shortest path distance or any other similarity measure. While our method is able to closely approximate quite different similarity measures as we show

[1] https://github.com/uhh-lt/path2vec

on WordNet-based measures and therefore can be used in lieu of these measures in NLP components and applications, our main point is the increase of speed in the similarity computation of nodes, which gains up to 4 orders of magnitude with respect to the original graph-based algorithms.

1 Graph Metric Embeddings Model

Definition of the Model *Path2vec* learns embeddings of the graph nodes $\{v_i, v_j\} \in V$ such that the dot products between pairs of the respective vectors $(\mathbf{v}_i \cdot \mathbf{v}_j)$ are close to the user-defined similarities between the nodes s_{ij}. In addition, the model reinforces the similarities $\mathbf{v}_i \cdot \mathbf{v}_n$ and $\mathbf{v}_j \cdot \mathbf{v}_m$ between the nodes v_i and v_j and all their respective adjacent nodes $\{v_n : \exists(v_i, v_n) \in E\}$ and $\{v_m : \exists(v_j, v_m) \in E\}$ to preserve local structure of the graph. The model preserves both global and local relations between nodes by minimizing $\sum_{(v_i,v_j)\in B}((\mathbf{v}_i^\top \mathbf{v}_j - s_{ij})^2 - \alpha(\mathbf{v}_i^\top \mathbf{v}_n + \mathbf{v}_j^\top \mathbf{v}_m))$, where $s_{ij} = sim(v_i, v_j)$ is the value of a 'gold' similarity measure between a pair of nodes v_i and v_j, $\mathbf{v}_i$ and $\mathbf{v}_j$ are the embeddings of the first and the second node, B is a training batch, α is a regularization coefficient. The second term $(\mathbf{v}_i \cdot \mathbf{v}_n + \mathbf{v}_j \cdot \mathbf{v}_m)$ in the objective function is a regularizer that aids the model to simultaneously maximize the similarity between adjacent nodes while learning the similarity between the two target nodes (one adjacent node is randomly sampled for each target node).

We use negative sampling to form a training batch B adding p negative samples ($s_{ij} = 0$) for each real ($s_{ij} > 0$) training instance: each real node (synset) pair (v_i, v_j) with 'gold' similarity s_{ij} is accompanied with p 'negative' node pairs (v_i, v_k) and (v_j, v_l) with zero similarities, where v_k and v_l are randomly sampled nodes from V. Embeddings are initialized randomly and trained using the *Adam* optimizer (Kingma and Ba, 2015) with early stopping. Once the model is trained, the computation of node similarities is approximated with the dot product of the learned node vectors, making the computations efficient: $\hat{s}_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j$.

Relation to Similar Models Our model bears resemblance to the Skip-gram model (Mikolov et al., 2013), where the vector dot product $\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j$ of vectors of pairs of words (v_i, v_j) from a training corpus is optimized to a high score close to 1 for observed samples, while the dot products of negative samples are optimized towards 0. In the Skip-gram model, the target is to minimize the log likelihood of the conditional probabilities of context words w_j given current words w_i: $\mathcal{L} = -\sum_{(v_i,v_j)\in B_p} \log \sigma(\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j) - \sum_{(v_i,v_j)\in B_n} \log \sigma(-\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j)$, where B_p is the batch of positive training samples, B_n is the batch of the generated negative samples, and σ is the sigmoid function. At this, Skip-gram uses only local information, never creating the full co-occurrence count matrix. In our *path2vec* model, the target dot product values s_{ij} are not binary, but can take arbitrary values in the $[0...1]$ range, as given by the custom distance metric. Further, we use only a single embedding matrix with vector representations of the graph nodes, not needing to distinguish target and context.

Another related model is Global Vectors (GloVe) (Pennington et al., 2014), which learns co-occurrence probabilities in a given corpus. The objective function to be minimized in GloVe model is $\mathcal{L} = \sum_{(v_i,v_j)\in B} f(s_{ij})(\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j - \log s_{ij} + b_i + b_j)^2$, where s_{ij} counts the co-occurrences of words v_i and v_j, b_i and b_j are additional biases for each word, and $f(s_{ij})$ is a weighting function handling rare co-occurrences. Like the Skip-gram, GloVe also uses two embedding matrices, but it relies only on global information, pre-aggregating global word co-occurrence counts.

Computing Training Similarities In general case, our model requires computing pairwise node similarities s_{ij} for training between all pairs of nodes in the input graph G. This step could be computationally expensive, but it is done only once to make computing of similarities fast. Besides, for some metrics, effective algorithms exist that compute all pairwise similarities at once, e.g. Johnson (1977) algorithm for computing shortest paths distances with the worst-case performance of $O(|V|^2 \log |V| + |V||E|)$. As the input training dataset also grows quadratically in $|V|$, training time for large graphs can be slow. To address this issue, we found it useful to prune the input training set so that each node $v_i \in V$ has only $k \in [50; 200]$ most similar nodes. Such pruning does not lead to loss of effectiveness.

2 Computational Efficiency

Experimental Setting In this section, we compare efficiency of our method as compared to the original graph based similarity metrics. We

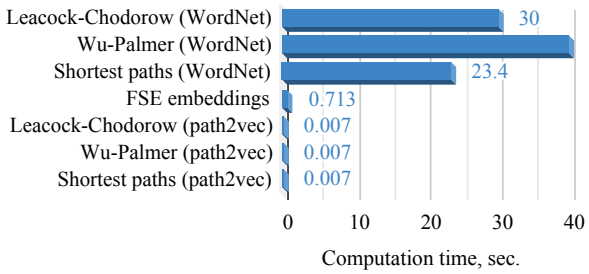

Figure 1: Similarity computation: graph vs vectors.

trained the model on a graph of 82,115 noun synsets from WordNet. Using NLTK (Bird et al., 2009) we computed the following metrics: (1) Leacock-Chodorow similarities (*LCH*) based on the shortest path between two synsets in the WordNet hypernym/hyponym taxonomy and its maximum depth; (2) inverted shortest path distance (*ShP*); (3) Wu-Palmer similarities (*WuP*) based on the depth of the two nodes in the taxonomy and the depth of their most specific ancestor node. For instance, for *LCH* this procedure took about 30 hours on an Intel Xeon E5-2603v4@1.70GHz CPU using 10 threads. We pruned similarities to the first 50 most similar 'neighbors' of each synset and trained *path2vec* on this dataset.

Discussion of Results Figure 1 presents computation times for pairwise similarities between one synset and all other 82,115 WordNet noun synsets. We compare running times of calculating two original graph-based metrics to Hamming distance between 128D FSE binary embeddings (Subercaze et al., 2015) and to dot product between their dense vectorized 300D counterparts (using CPU). Using float vectors (*path2vec*) is 4 orders of magnitude faster than operating directly on graphs, and 2 orders faster than Hamming distance. The dot product computation is much faster as compared to shortest path computation (and other complex walks) on a large graph. Also, low-dimensional vector representations of nodes take much less space than the pairwise similarities between all the nodes. The time complexity of calculating the shortest path between graph nodes (as in *ShP* or *LCH*) is in the best case linear in the number of nodes and edges. Calculating Hamming distance between binary strings is linear in the sum of string lengths, which are equivalent of vector sizes (Hamming, 1950). At the same time, the complexity of calculating dot product between float vectors is linear in the vector size and is easily parallelized.

	LCH	ShP	WuP	LCH	ShP	WuP
WordNet	*100*	*100*	*100*	51.3	51.3	47.4
path2vec	**93.5**	**95.2**	**93.1**	**53.2**	**55.5**	**55.5**
TransR	77.6	77.6	72.5		38.6	
node2vec	75.9	75.9	78.7		46.2	
DeepWalk	86.8	86.8	85.0		53.3	
FSE	90.0	90.0	89.0		55.6	

Table 1: Spearman correlations with WordNet similarities (left) and human judgments (right) $\times 100$.

3 Evaluation on Semantic Similarity

Experimental Setting We use noun pairs from the SimLex999 dataset (Hill et al., 2015), measuring Spearman rank correlation between 'gold' WordNet distances for these pairs and the vector distances produced by the graph embedding models (trained on WordNet) to see how well the models fit the training objective. We also test the plausibility of the model's output to human judgments. For this, we use human-annotated similarities from the same SimLex999. Some SimLex999 lemmas can be mapped to more than one WordNet synset. We chose the synset pair with the highest dot product between the embeddings from the corresponding model.

Baselines Our model is compared against five baselines: *raw WordNet similarities* by respective measures; *DeepWalk* (Perozzi et al., 2014); *node2vec* (Grover and Leskovec, 2016); *FSE* (Subercaze et al., 2015); *TransR* (Lin et al., 2015). *DeepWalk*, *node2vec*, and *TransR* models were trained on the same WordNet graph. We used all 82,115 noun synsets as vertices and hypernym/hyponym relations between them as edges. During the training of *DeepWalk* and *node2vec* models, we tested different values for the number of random walks (in the range from 10 to 100), and the vector size (100 to 600). For *DeepWalk*, we additionally experimented with the window size (5 to 100). All other hyperparameters were left at default values. *FSE* embeddings of the WordNet noun synsets were provided to us by the authors, and consist of 128-bit vectors.

Discussion of Results The left part of Table 1 shows results with the WordNet similarity scores used as gold standard. *Path2vec* outperforms other graph embeddings, achieving high correlations with WordNet similarities. This shows that our model efficiently approximates different graph measures. The right part of Table 1 shows results

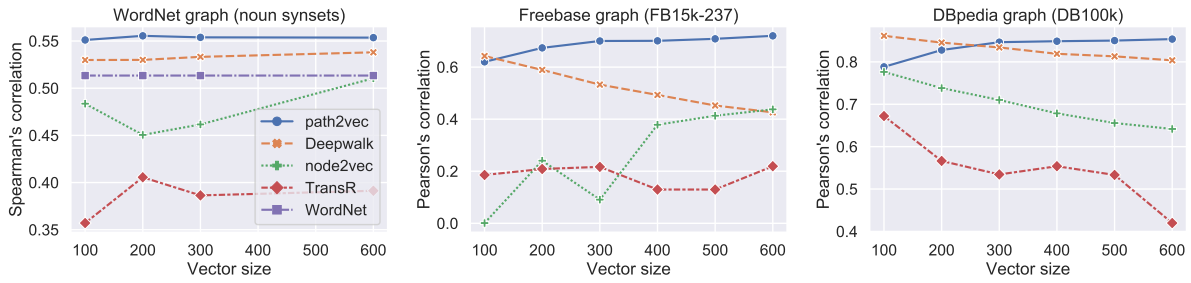

Figure 2: Evaluation on different graphs on SimLex999 (left) and shortest path distance (middle, right).

for the correlations with human judgments (SimLex999). We report the results for the best models for each method, all of them (except *FSE*) using vector size 300 for comparability.

Figure 2 (left) compares *path2vec* to the baselines, as measured by the correlations with SimLex999 human judgments. The WordNet line denotes the correlation of WordNet similarities with SimLex999 scores. For the *path2vec* models, there is a tendency to improve the performance when the vector size is increased (horizontal axis), until a plateau is reached beyond 600. Note that *node2vec* fluctuates, yielding low scores for 200 dimensions. The reported best *DeepWalk* models were trained with the 10 walks and window size 70. The reported best *node2vec* models were trained with 25 walks. Interestingly, *path2vec* and *DeepWalk* models consistently *outperform* the raw WordNet.

4 Evaluation inside a WSD Algorithm

Experimental Setting To showcase how our approach can be be used inside a graph-based algorithm, we employ word sense disambiguation (WSD) task, reproducing the approach of (Sinha and Mihalcea, 2007). We replace graph similarities with the dot product between node embeddings and study how it influences the WSD performance. The WSD algorithm starts with building a graph where the nodes are the WordNet synsets of the words in the input sentence. The nodes are then connected by edges weighted with the similarity values between the synset pairs. The final step is selecting the most likely sense for each word based on the weighted in-degree centrality score for each synset.

Discussion of Results Table 2 presents the WSD micro-F1 scores using raw WordNet similarities, 300D *path2vec*, *DeepWalk* and *node2vec* models, and the 128D *FSE* model. We evaluate on the following all-words English WSD test sets:

Model	**Senseval2**	**Senseval3**	**SemEval-15**
Random sense	0.381	0.312	0.393
Graph-based vs vector-based measures			
LCH (WordNet)	0.547	0.494	0.550
LCH (path2vec)	0.527$_{\downarrow 0.020}$	0.472$_{\downarrow 0.022}$	0.536$_{\downarrow 0.014}$
ShP (WordNet)	0.548	0.495	0.550
ShP (path2vec)	0.534$_{\downarrow 0.014}$	0.489$_{\downarrow 0.006}$	0.563$_{\uparrow 0.013}$
WuP (WordNet)	0.547	0.487	0.542
WuP (path2vec)	0.543$_{\downarrow 0.004}$	0.489$_{\uparrow 0.002}$	0.545$_{\uparrow 0.003}$
Various baseline graph embeddings trained on WordNet			
TransR	0.540	0.466	0.536
node2vec	0.503	0.467	0.489
DeepWalk	0.528	0.476	0.552
FSE	0.536	0.476	0.523

Table 2: F1 scores of a graph-based WSD algorithm on WordNet versus its vectorized counterparts.

Senseval-2 (Palmer et al., 2001), Senseval-3 (Mihalcea et al., 2004), and SemEval-15 Task 13 (Moro and Navigli, 2015). The raw WordNet similarities have a small edge over their vector approximations in the majority of the cases yet the *path2vec* models consistently closely follow them while outperforming other graph embedding baselines: We indicate the differences with respect to the original with a subscript number.

5 Evaluation on Knowledge Base Graphs

5.1 Experimental Settings

To show the utility of our model besides the WordNet graph, we also applied it to two graphs derived from knowledge bases (KBs). More specifically, we base our experiments on two publicly available standard samples from these two resources: the FB15k-237 (Toutanova and Chen, 2015) dataset contains 14,951 entities/nodes and is derived from Freebase (Bollacker et al., 2008); the DB100k (Ding et al., 2018) dataset contains 99,604 entities/nodes and is derived from DBPe-

dia (Auer et al., 2007).

It is important to note that both datasets were used to evaluate approaches that learn knowledge graph embeddings, e.g. (Lin et al., 2015; Xie et al., 2016; Joulin et al., 2017) on the task on *knowledge base completion* (KBC), to predict missing KB edges/relations between nodes/entities. The specificity of our model is that it learns a given graph similarity metric, which is not provided in these datasets. Therefore, we use only the graphs from these datasets, computing the shortest path distances between all pairs of nodes using the algorithm of Johnson (1977). Instead of the KBC task, we evaluate on the task of predicting node similarity, here using the shortest path distance. We generate a random sample of node pairs for testing from the set of all node pairs (these pairs are excluded from training). The test set contains an equal number of paths of length 1-7 (in total 1050 pairs each, 150 pairs per path length).

5.2 Discussion of Results

Figure 2 (middle and right) shows evaluation results on the knowledge base graphs. *Path2vec* is able to better approximate the target graph metric than the standard graph embedding models. As dimensionality of the embeddings increases, the model more closely approximates the target metric, but the performance drop for the models with a low number of dimensions is not drastic, allowing more effective computations while maintaining a reasonable efficiency level. Regarding the competitors, DeepWalk comes closest to the performance of our approach, but does not seem to make use of the additional dimensions when training on larger vector sizes; on the DBPedia dataset, this issue is shared between all baselines, where correlation to the true path lengths decreases as representation length increases.

6 Related Work

Representation learning on graphs received much attention recently in various research communities, see Hamilton et al. (2017a) for a thorough survey on the existing methods. All of them (including ours) are based on the idea of projecting graph nodes into a latent space with a much lower dimensionality than the number of nodes. Existing approaches to graph embeddings use either factorization of the graph adjacency matrix (Cao et al., 2015; Ou et al., 2016) or random walks over the graph as in *Deepwalk* (Perozzi et al., 2014) and *node2vec* (Grover and Leskovec, 2016). A different approach is taken by Subercaze et al. (2015), who directly embed the WordNet tree graph into Hamming hypercube binary representations. Their 'Fast similarity embedding' (*FSE*) model provides a quick way of calculating semantic similarities based on WordNet. The *FSE* embeddings are not differentiable though, considerably limiting their use in deep neural architectures. *TransR* (Lin et al., 2015) extends *TransH* (Wang et al., 2014) and is based on the idea that an entity may have a few aspects and different relations are focused on them. So the same entities can be close or far from each other depending on the type of the relation. *TransR* projects entity vectors into a relation specific space, and learns embeddings via translation between projected entities.

We compare our *path2vec* model to these approaches, yet we did not compare to the models like *GraphSAGE* embeddings (Hamilton et al., 2017b) and Graph Convolutional Networks (Schlichtkrull et al., 2018) as they use node features which are absent in our setup.

7 Conclusion

Structured knowledge contained in language networks is useful for NLP applications but is difficult to use directly in neural architectures. We proposed a way to train embeddings that directly represent a graph-based similarity measure structure. Our model, *path2vec*, relies on both global and local information from the graph and is simple, effective, and computationally efficient. We demonstrated that our approach generalizes well across graphs (WordNet, Freebase, and DBpedia). Besides, we integrated it into a graph-based WSD algorithm, showing that its vectorized counterpart yields comparable F1 scores on three datasets.

Path2vec enables a speed-up of up to four orders of magnitude for the computation of graph distances as compared to 'direct' graph measures. Thus, our model is simple and general, hence it may be applied to any graph together with a node distance measure to speed up algorithms that employ graph distances.

Acknowledgments

This was supported by the DFG under "JOIN-T" (BI 1544/4) and "ACQuA" (BI 1544/7) projects.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web: Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, pages 722–735, Busan, South Korea. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, Vancouver, BC, Canada. ACM.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900, Melbourne, Australia. ACM.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 110–121, Melbourne, Australia. Association for Computational Linguistics.

Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355–369.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, San Francisco, CA, USA. ACM.

William Hamilton, Rex Ying, and Jure Leskovec. 2017a. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74.

William Hamilton, Zhitao Ying, and Jure Leskovec. 2017b. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, Long Beach, CA, USA.

Richard Hamming. 1950. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, 41(4):665–695.

Donald B. Johnson. 1977. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. 2017. Fast linear model for knowledge graph embeddings. *arXiv preprint arXiv:1710.10881*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Andrey Kutuzov, Mohammad Dorgham, Oleksiy Oliynyk, Chris Biemann, and Alexander Panchenko. 2019. Learning graph embeddings from WordNet-based similarity measures. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 125–135, Minneapolis, MN, USA. Association for Computational Linguistics.

Bertrand Lebichot, Guillaume Guex, Ilkka Kivimäki, and Marco Saerens. 2018. A constrained randomized shortest-paths framework for optimal exploration. *arXiv preprint arXiv:1807.04551*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187, Austin, TX, USA. AAAI Press.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28, Barcelona, Spain. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, Lake Tahoe, NV, USA. Curran Associates, Inc.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297. Association for Computational Linguistics.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, San Francisco, CA, USA. ACM.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 21–24, Toulouse, France. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, New York, NY, USA. ACM.

Mohammad T. Pilehvar and Roberto Navigli. 2015. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the European Semantic Web Conference 2018: The Semantic Web*, pages 593–607, Heraklion, Greece. Springer.

Ravi Sinha and Rada Mihalcea. 2007. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *International Conference on Semantic Computing (ICSC)*, pages 363–369, Irvine, CA, USA. IEEE.

Julien Subercaze, Christophe Gravier, and Frédérique Laforest. 2015. On metric embedding for boosting semantic similarity computations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 8–14, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119, Québec City, QC, Canada. AAAI Press.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2659–2665, Phoenix, AZ, USA. AAAI Press.