Improving Neural Entity Disambiguation with Graph Embeddings

Özge Sevgili[†], **Alexander Panchenko**[‡, †, ⋆], and **Chris Biemann**[†]

[†]Universität Hamburg, Hamburg, Germany
[‡]Skolkovo Institute of Science and Technology, Moscow, Russia
[⋆]Diffbot Inc., Menlo Park, CA, USA
{sevgili,panchenko,biemann}@informatik.uni-hamburg.de

Abstract

Entity Disambiguation (ED) is the task of linking an ambiguous entity mention to a corresponding entry in a knowledge base. Current methods have mostly focused on unstructured text data to learn representations of entities, however, there is structured information in the knowledge base itself that should be useful to disambiguate entities. In this work, we propose a method that uses graph embeddings for integrating structured information from the knowledge base with unstructured information from text-based representations. Our experiments confirm that graph embeddings trained on a graph of hyperlinks between Wikipedia articles improve the performances of simple feed-forward neural ED model and a state-of-the-art neural ED system.

1 Introduction

The inherent and omnipresent ambiguity of language at the lexical level results in ambiguity of words, named entities, and other lexical units. Word Sense Disambiguation (WSD) (Navigli, 2009) deals with individual ambiguous words such as nouns, verbs, and adjectives. The task of Entity Linking (EL) (Shen et al., 2015) is devoted to the disambiguation of mentions of named entities such as persons, locations, and organizations. Basically, EL aims to resolve such ambiguity by creating an automatic reference between an ambiguous entity mention/span in a context and an entity in a knowledge base. These entities can be Wikipedia articles and/or DBpedia (Mendes et al., 2011)/Freebase (Bollacker et al., 2008) entries. EL can be divided into two subtasks: (i) Mention Detection (MD) or Name Entity Recognition (NER) (Nadeau and Sekine, 2007) finds entity references from a given raw text; (ii) and Entity Disambiguation (ED) assigns entity references for a given mention in context. This work deals with the entity disambiguation task.

The goal of an ED system is resolving the ambiguity of entity mentions, such as *Mars, <u>Galaxy</u>, and Bounty are all delicious*. It is hard for an algorithm to identify whether the entity is an astronomical structure[1] or a brand of milk chocolate[2].

Current neural approaches to EL/ED attempt to use context and word embeddings (and sometimes entity embeddings on mentions in text) (Kolitsas et al., 2018; Sun et al., 2015). Whereas these and most other previous approaches employ embeddings trained from text, we aim to create entity embeddings based on structured data (i.e. hyperlinks) using graph embeddings and integrate them into the ED models.

Graph embeddings aim at representing nodes in a graph, or subgraph structure, by finding a mapping between a graph structure and the points in a low-dimensional vector space (Hamilton et al., 2017). The goal is to preserve the features of the graph structure and map these features to the geometric relationships, such as distances between different nodes, in the embedding space. Using fixed-length dense vector embeddings as opposed to operating on the knowledge bases' graph structure allows the access of the information encoded in the graph structure in an efficient and straightforward manner in modern neural architectures.

Our claim is that including graph structure features of the knowledge base has a great potential to make an impact on ED. In our first experiment, we present a method based on a simple neural network with the inputs of a context, entity mention/span, explanation of a candidate entity, and a candidate entity. Each entity is represented by graph embeddings, which are created using the knowledge base, DBpedia (Mendes et al., 2011)

[1]http://dbpedia.org/resource/Galaxy
[2]http://dbpedia.org/resource/Galaxy_(chocolate)

containing hyperlinks between entities. We perform ablation tests on the types of inputs, which allows us to judge the impact of the single inputs as well as their interplay. In a second experiment, we enhance a state-of-the-art neural entity disambiguation system called end2end (Kolitsas et al., 2018) with our graph embeddings: The original system relies on character, word and entity embeddings; we replace respectively complement these with our graph embeddings. Both experiments confirm the hypothesis that structured information in the form of graph embeddings are an efficient and effective way of improving ED.

Our **main contribution** is a creation of a simple technique for integration of structured information into an ED system with graph embeddings. There is no obvious way to use large structured knowledge bases directly in a neural ED system. We provide a simple solution based on graph embeddings and confirm experimentally its effectiveness.

2 Related Work

Entity Linking Traditional approaches to EL focus on defining the similarity measurement between a mention and a candidate entity (Mihalcea and Csomai, 2007; Strube and Ponzetto, 2006; Bunescu and Paşca, 2006). Similarly, Milne and Witten (2008) define a measurement of entity-entity relatedness. Current state-of-the-art approaches are based on neural networks (Huang et al., 2015; Ganea and Hofmann, 2017; Kolitsas et al., 2018; Sun et al., 2015), where are based on character, word and/or entity embeddings created by a neural network with a motivation of their capability to automatically induce features, as opposed to hand-crafting them. Then, they all use these embeddings in neural EL/ED.

Yamada et al. (2016) and Fang et al. (2016) utilize structured data modelling entities and words in the same space and mapping spans to entities based on the similarity in this space. They expand the objective function of word2vec (Mikolov et al., 2013a,b) and use both text and structured information. Radhakrishnan et al. (2018) extend the work of Yamada et al. (2016) by creating their own graph based on co-occurrences statistics instead of using the knowledge graph directly. Contrary to them, our model learns a mapping of spans and entities, which reside in different spaces and use graph embeddings trained on the knowledge graph for representing structured information.

Kolitsas et al. (2018) address both MD and ED in their end2end system. They build a context-aware neural network based on character, word, and entity embeddings coupled with attention and global voting mechanisms. Their entity embeddings, proposed by Ganea and Hofmann (2017), are computed by the empirical conditional word-entity distribution based on the co-occurrence counts on Wikipedia pages and hyperlinks.

Graph Embeddings There are various methods to create graph embedding, which can be grouped into the methods based on matrix factorization, random walks, and deep learning (Goyal and Ferrara, 2018). Factorization-based models depend on the node adjacency matrix and dimensionality reduction method (Belkin and Niyogi, 2001; Roweis and Saul, 2000; Tang et al., 2015). Random-walk-based methods aim to preserve many properties of graph (Perozzi et al., 2014; Grover and Leskovec, 2016). Deep-learning-based ones reduce dimensionality automatically and model non-linearity (Wang et al., 2016; Kipf and Welling, 2017). In our case, efficiency is crucial and time complexity of factorization-based models is high. The disadvantage of the deep-learning-based models is that they require extensive hyperparameter optimization. To keep it simple, efficient, and to minimize the numbers of hyperparameters to tune, yet still effective, we select random-walk-based methods, where two prominent representatives are DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016).

3 Learning Graph-based Entity Vectors

In order to make information from a semantic graph available for an entity linking system, we make use of graph embeddings. We use DeepWalk (Perozzi et al., 2014) to create the representation of entities in the DBPedia. DeepWalk is scalable, which makes it applicable on a large graph. It uses random walks to learn latent representations and provides a representation of each node on the basis of the graph structure.

First, we created a graph whose nodes are unique entities; attributes are explanations of entities, i.e. long abstracts; edges are the page links between entities with the information from DBpedia. Second, a vector representation per entity is generated by training DeepWalk on the edges of this graph. For this, we used all default hyper-parameters of DeepWalk, e.g. *number-*

Entity	Most similar 3 entities
Michael_Jordan_(basketball)	Charles_Barkley, Scottie_Pippen, Larry_Bird
Michael_I._Jordan	David_Blei, Machine_learning , Supervised_learning
Michael_Jordan_(footballer)	Dagenham_&_Redbridge_F.C., Stevenage_F.C., Yeovil_Town_F.C.

Table 1: **Graph entity embeddings:** Top three most similar entities for the name "Michael Jordan" based on our 400-dimensional DeepWalk embeddings.

walks is 10, *walk-length* is 40, and *window-size* is 5. To exemplify the result, the most similar 3 entities of disambiguated versions of Michael_Jordan, in the trained model with 400-dimension vectors are shown in Table 1. The first entity, Michael_Jordan_(basketball), is a well-known basketball player, and his all most similar entities are all basketball players and of similar age. The second entity, Michael_I._Jordan is a scientist, and again the most similar entities are either scientists in the same field or the topics of his study field. The last entity, Michael_Jordan_(footballer), is a football player whose most similar entities are football clubs. This suggests that our graph entity embeddings can differentiate different entities with the same name.

4 Experiment 1: Entity Disambiguation with Text and Graph Embeddings

In our first experiment, we build a simple neural ED system based on a feed-forward network and test the utility of the graph embeddings as compared to text-based embeddings.

4.1 Description of the Neural ED Model

The inputs of an ED task are a context and a possibly ambiguous entity span, and the output is a knowledge base entry. For example, *Desire contains a duet with Harris in the song Joey* and *Desire* given as an input and the output is *Bob Dylan's album* entity[3].

Our model in this experiment is a feed-forward neural network. Its input is a concatenation of document vectors of a context, a span, and an explanation of the candidate entity, i.e. long abstract, and graph embedding of a candidate entity as in Figure 1, and output is a prediction value denoting whether the candidate entity is correct in this context. For learning representations, we employ doc2vec (Le and Mikolov, 2014) for text and DeepWalk (Perozzi et al., 2014) for graphs, both methods have shown good performance on other tasks. We will describe the input components in more detail in the following.

[3]http://dbpedia.org/page/Desire_(Bob_Dylan_album)

Creating Negative Samples: It is not computationally efficient to use all entities in our graph as a candidate for every context-span as negative examples for training because of the high number of entities (about 5 million). Thus, we need to filter some possible entities for each context-span in order to generate negative samples. We use spans to find out possible entities. If any lemma in the span is contained in an entity's name, the entity is added to the candidates for this mention. For example, if the span is *undergraduates*, the entity Undergraduate_degree is added to the candidates.

For training, we generate negative samples by filtering this candidate list and limited the number of candidates per positive sample. We employ two techniques to filter the candidate list. First, we shuffle the candidate list and randomly select *n* candidates. The other is to select the closest candidates by the following score formula: $score = \frac{\#\ of\ intersection \times page\ rank}{length}$, where $\#\ of\ intersection$ means the number of the common words between span/entity mention and candidate entity, $page\ rank$ is the page rank value (Page et al., 1999) on the entire graph for the candidate entity, and length is the number of tokens in the entity's name/title, e.g. the length of the entity Undergraduate_degree is 2. Before taking candidates with highest *n* scores, we have pruned the most similar candidates to the correct entity on the basis of the cosine between their respective graph embeddings. The reason for pruning is to assure that the entities are distinctive enough from each other so that a classifier can learn the distinction.

Word and Context Vectors: Document embedding techniques like doc2vec (Le and Mikolov, 2014) assign each document a single vector, which gets adjusted with respect to all words in the document and all document vectors in the dataset. Additionally, doc2vec provides the *infer_vector* method, which takes a word sequence and returns its representation. We employ this function for representing contexts (including the entity span),

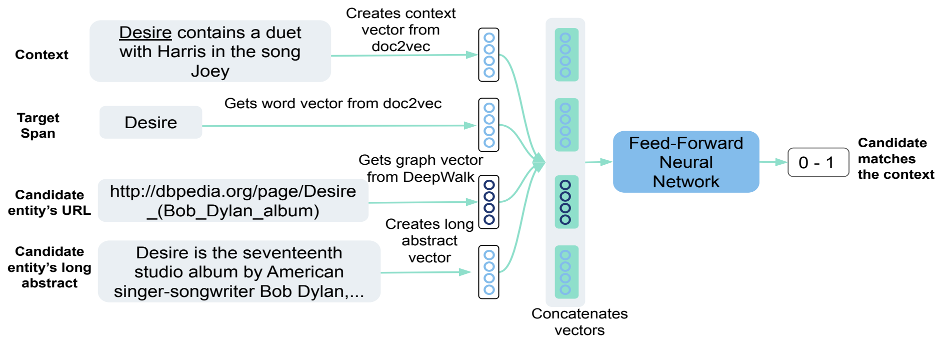
Context
Desire contains a duet with Harris in the song Joey
Creates context vector from doc2vec
Target Span
Desire
Gets word vector from doc2vec
Candidate entity's URL
http://dbpedia.org/page/Desire_(Bob_Dylan_album)
Gets graph vector from DeepWalk
Candidate entity's long abstract
Desire is the seventeenth studio album by American singer-songwriter Bob Dylan,...
Creates long abstract vector
Concatenates vectors
Feed-Forward Neural Network
0 - 1
Candidate matches the context

Figure 1: **Architecture of our feed-forward neural ED system:** using Wikipedia hyperlink graph embeddings as an additional input representation of entity candidates.

entity explanations (long abstracts), and multi-word spans.

4.2 Experimental Setup

Datasets: An English Wikipedia 2017 dump has been used to train doc2vec, using the gensim implementation (Řehůřek and Sojka, 2010). There are about 5 million entities (nodes), and 112 million page links (edges), in our graph.

DBpedia Spotlight (Mendes et al., 2011) (331 entities), *KORE50* (Hoffart et al., 2012) (144 entities), and *Reuters-128* (Röder et al., 2014) (881 entities) datasets as described in (Rosales-Méndez et al., 2018) are used to train and test our architecture. We have used 80% of these data for training, 10% for development, and the remaining for testing.

Implementation Details: We fixed context, span, and long abstract embedding dimensionality to 100, the default parameter defined in the implementation of gensim (Řehůřek and Sojka, 2010). The size of the graph embeddings is 400. We optimize the graph embedding size based on the development set with the range $100 - 400$. The overall input size is 700 when concatenating context, span, long abstract, and graph entity embeddings.

The number of negative samples per positive sample is 10. We have 3 hidden layers with equal sizes of 100. In the last layer, we have applied the *tanh* activation function. We have used *Adam* (Kingma and Ba, 2014) optimizer with a learning rate of 0.005 and 15000 epochs. All hyper-parameters are determined by preliminary experiments.

4.3 Evaluation

The evaluation shows the impact of graph embeddings in a rather simple learning architecture.

In this experiment, an ablation test is performed to analyze the effect of graph embeddings. We have two types of training sets, where the creation of negative samples differs (in one of them, we have filtered negative samples randomly, whereas, in the other, we filtered them by selecting the closest ones, as explained in Section 4.1). In Figure 2, the upper part shows the Accuracy, Precision, Recall, and F1 values of the training set filtered randomly while the lower part results refer to the training set filtered by selecting closest neighbors. The first bar in the charts contains the result of the input, which concatenates context and long abstract embeddings (in this condition the input size becomes 200), here entity information only comes from its long abstract. The second bar presents the results of the input combination, context, word/span, and long abstract embeddings (the size of the input is 300). In the third bar, the input is the concatenation of context, long abstract, and graph embeddings (the input size is 600). Finally, the last bar indicates results for the concatenation of all types of inputs, for an input size of 700. For each configuration, we run the model 5 times and get the mean and standard deviation values. In Figure 2, charts show the mean values and the lines on the charts indicate standard deviation.

Comparing the first and third bars (or the second and last bars) in Figure 2, we can clearly see the results are increased when the input includes the graph embeddings for both variants of negative sampling. Comparing the third and last bars (or the first and second bars), we observe that including the span representation slightly decreases results for both sampling variants. We attribute this to the presence of the context embedding, which already includes the span, thus this increases the number of parameters of the network without sub-

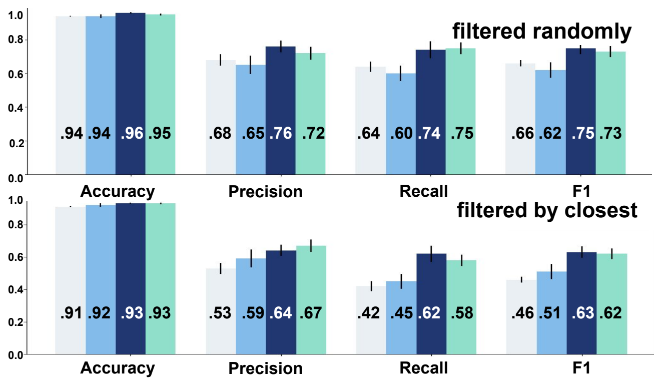

Figure 2: **Entity disambiguation performance**: various representations of our neural feed-forward ED system (cf. Figure 1). Reported are scores on the positive class filtered randomly and closest neighbors: ⬜ context+long abstract, 🟦 context+long abstract+span, ■ context+long abstract+graph, 🟩 context+long abstract+span+graph.

stantially adding new information. Appending the graph embeddings improves the results about $0.09-0.17$ in F1, $0.13-0.2$ in recall, $0.07-0.12$ in precision and $0.01-0.02$ in accuracy scores. In general, the randomly sampled dataset is easier as it contains less related candidates.

5 Experiment 2: Integrating Graph Embeddings in the end2end ED System

5.1 Description of the Neural ED Model

For the second experiment, we have used the end2end state-of-the-art system for EL/ED (Kolitsas et al., 2018) and expanded it with our graph embeddings. In this neural end-to-end entity disambiguation system, standard text-based entity embeddings are used. In the experiment described in this section, we replace or combine them (keeping the remaining architecture unchanged) with our graph embeddings build as described in Section 3.

We replaced end2end's entity vector with our graph embeddings and the concatenation of their entity vector and our graph embeddings. We use the GERBIL (Usbeck et al., 2015) benchmark platform for an evaluation.

5.2 Experimental Setup

Datasets: We train the neural end2end system in its default configuration with the combination of *MSNBC* (Cucerzan, 2007) (747 entities), *ACE2004* (Ratinov et al., 2011) (306 entities), *AQUAINT* (Ratinov et al., 2011) (727 entities), *ClueWeb*, and *Wikipedia* datasets. We test the system on the GERBIL (Usbeck et al., 2015) platform using *DBpedia Spotlight* (Mendes et al., 2011) (331 entities) and *Reuters-128* (Röder et al., 2014) (881 entities) datasets.

Implementation Details: We have not changed hyper-parameters for training the end2end system[4] (We used their base model + global for ED setting). We create graph embeddings with the same technique used before, however, to keep everything the same, we decided to also use 300 dimensions for the graph embeddings in this experiment to match the dimensionality of end2end's space.

We create the embeddings file with the same format they used. They give an id for each entity and call it "wiki id". First, we generate a map between this wiki id and our graph id (id of our entity). Then, we replace each entity vector corresponding to the wiki id with our graph embeddings, which refers to the entity. Sometimes there is no corresponding graph entity for the entity in the end2end system, in this case, we supply a zero vector.

They have a stopping condition, which applies after 6 consecutive evaluations with no significant improvement in the Macro F1 score. We have changed this hyperparameter to 10, accounting for our observation that the training converges slower when operating on graph embeddings.

5.3 Evaluation

Table 2 reports ED performance evaluated on *DBpedia Spotlight* and *Reuters-128* datasets. There are three models, end2end trained using their text entity vectors, our graph embeddings and the combination of them. Training datasets and implementation details are the same for all models. We train

[4]https://github.com/dalab/end2end_neural_el

DBpedia Spotlight dataset						
Model	Macro F1	Macro Precision	Macro Recall	Micro F1	Micro Precision	Micro Recall
text embeddings	0.762	0.790	0.742	0.781	0.815	0.750
graph embeddings	0.796	**0.860**	0.758	0.783	**0.847**	0.730
text and graph embeddings	**0.798**	0.835	**0.775**	**0.797**	0.835	**0.763**
Reuters-128 dataset						
Model	Macro F1	Macro Precision	Macro Recall	Micro F1	Micro Precision	Micro Recall
text embeddings	0.593	0.654	0.575	0.634	0.687	0.589
graph embeddings	0.607	**0.694**	0.574	**0.660**	**0.747**	0.592
text and graph embeddings	**0.614**	0.687	**0.590**	0.650	0.707	**0.602**

Table 2: **Entity disambiguation performance**: The end2end (Kolitsas et al., 2018) system based on the original text-based embeddings, our graph embeddings and a combination of both evaluated using the GERBIL platform on *DBpedia Spotlight* and *Reuters-128* datasets.

the models for 10 times and removed the models that did not converge (1 non-converging run for each single type of embedding and 2 for the combination). Table 2 shows the mean values. The standard deviations of the models are between $0.02 - 0.05$ in the *DBpedia Spotlight* dataset and $0.01 - 0.03$ in the *Reuters-128* dataset over all scores. Scores are produced using the GERBIL platform; these are Micro-averaged over the set of annotations in the dataset and Macro-averaged over the average performance per document. The results are improved by including graph embeddings. When we compare two models, trained by graph embeddings and trained by entity vectors, the results are improved up to 0.03 in Macro F1 scores and Micro Precision, and up to 0.07 in Macro Precision. However, the improvement of the combination model is higher in Macro F1 and Recall. Micro-averaged results follow a similar trend. When we look at the scores of *Reuters-128* (Röder et al., 2014) dataset, the combination model improves Macro F1 and Recall and Micro Recall up to 0.02, 0.015, and 0.013 respectively. In the Micro-averaged evaluation, the combination model scores slightly below the model using graph embeddings alone.

To summarize the evaluation, our graph embeddings alone already lead to improvements over the original text-based embeddings, and their combination is even more beneficial. This suggests that test-based and graph-based representations in fact encode somewhat complementary information.

6 Conclusion and Future Work

We have shown how to integrate structured information into the neural ED task using two different experiments. In the first experiment, we use a simple neural network to gauge the impact of different text-based and graph-based embeddings. In the second experiment, we replace respectively complemented the representation of candidate entities in the ED component of a state-of-the-art EL system. In both setups, we demonstrate that graph embeddings lead to en par or better performance. This confirms our research hypothesis that it is possible to use structured resources for modeling entities in ED tasks and the information is complementary to a text-based representation alone. Our code and datasets are available online[5].

For future work, we plan to examine graph embeddings on other relationships, e.g. taxonomic or otherwise typed relations such as works-for, married-with, and so on, generalizing the notion to arbitrary structured resources. It might make a training step on the distance measure depending on the relation necessary. On the disambiguation architecture, modeling such direct links could give rise to improvements stemming from the mutual disambiguation of entities as e.g. done in (Ponzetto and Navigli, 2010). We will explore ways to map them into the same space to reduce the number of parameters. In another direction, we will train task-specific sentence embeddings.

Acknowledgments

We thank the SRW mentor Matt Gardner and anonymous reviewers for their most useful feedback on this work. The work was partially supported by a Deutscher Akademischer Austauschdienst (DAAD) doctoral stipend and the DFG-funded JOIN-T project BI 1544/4.

[5]`https://github.com/uhh-lt/kb2vec`

References

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 585–591, Cambridge, MA, USA. MIT Press.

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*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, Trento, Italy. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic. Association for Computational Linguistics.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 260–269, Berlin, Germany. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151(1 July):78–94.

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*, KDD '16, pages 855–864, New York, NY, USA. ACM.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40:52–74.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 545–554, New York, NY, USA. ACM.

Hongzhao Huang, Larry P. Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *CoRR*, abs/1504.07678.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference for Learning Representations (ICLR), San Diego, CA, USA, 2015.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, (ICLR)*, Toulon, France.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA. ACM.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Workshop Proceedings of the International Conference on Learning Representations (ICLR)*. 2013, Scottsdale, AZ, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Lake Tahoe, NV, USA. Curran Associates Inc.

David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of the 17th ACM*

Conference on Information and Knowledge Management, CIKM '08, pages 509–518, New York, NY, USA. ACM.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

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*, KDD '14, pages 701–710, New York, NY, USA. ACM.

Simone P. Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1522–1531, Uppsala, Sweden. Association for Computational Linguistics.

Priya Radhakrishnan, Partha Talukdar, and Vasudeva Varma. 2018. ELDEN: Improved entity linking using densified knowledge graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1844–1853, New Orleans, LA, USA. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Portland, OR, USA. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. European Language Resources Association (ELRA).

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - A collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3529–3533, Reykjavik, Iceland. European Language Resources Association (ELRA).

Henry Rosales-Méndez, Aidan Hogan, and Barbara Poblete. 2018. VoxEL: A benchmark dataset for multilingual entity linking. In *International Semantic Web Conference (2)*, volume 11137 of *Lecture Notes in Computer Science*, pages 170–186, Monterey, CA, USA. Springer.

Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460.

Michael Strube and Simone P. Ponzetto. 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1419–1424, Boston, MA, USA. AAAI Press.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1333–1339, Buenos Aires, Argentina. AAAI Press.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. 2015. GERBIL: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1133–1143, Florence, Italy. International World Wide Web Conferences Steering Committee.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA. ACM.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany. Association for Computational Linguistics.