Proceedings of the 4th Workshop on Deep Learning for Knowledge Graphs co-located with International Semantic Web Conference 2021

Mehwish Alam[1], Davide Buscaldi[2], Michael Cochez[3], Francesco Osborne[4], Diego Reforgiato Recupero[5], Harald Sack[1]

[1] FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany
[2] Laboratoire d'Informatique de Paris Nord, France
[3] Vrije Universiteit Amsterdam, the Netherlands
[4] Knowledge Media Institute (KMi), The Open University, UK
[5] University of Cagliari, Italy

Organizing Committee

- **Mehwish Alam**, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany
- **Davide Buscaldi**, Universitè Paris 13, USPC, Paris, France
- **Michael Cochez**, Vrije University of Amsterdam, the Netherlands
- **Francesco Osborne**, Knowledge Media Institute (KMi), The Open University, UK
- **Diego Reforgiato Recupero**, University of Cagliari, Cagliari, Italy
- **Harald Sack**, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany

Program Committee

- **Achim Rettinger**, Trier University, Germany
- **Angelo Salatino**, Knowledge Media Institute, Open University, United Kingdom
- **Blerina Gkotse**, CERN, Switzerland.
- **Danilo Dessì**, FIZ-Karlsruhe, Karlsruhe Institute of Technology, Germany.
- **Femke Ongenae**, Ghent University, Belgium.
- **Finn Årup Nielsen**, Technical University of Denmark, Denmark.
- **Genet Asefa Gesese**, FIZ-Karlsruhe, Karlsruhe Institute of Technology, Germany.
- **Heiko Paulheim**, University of Mannheim, Germany.
- **Max Berrendorf**, LMU Munich, Germany.
- **Mayank Kejriwal**, University of Southern California, USA.
- **Peter Bloem**, Vrije Universiteit Amsterdam, the Netherlands.
- **Rima Türker**, FIZ-Karlsruhe, Karlsruhe Institute of Technology, Germany.
- **Russa Biswas**, FIZ-Karlsruhe, Karlsruhe Institute of Technology, Germany.
- **Thiviyan Thanapalasingam**, University of Amsterdam, the Netherlands.

Preface

Knowledge Graphs have been used in various machine learning tasks by deriving latent feature representations of entities and relations. Knowledge Graphs represent formal semantics by describing entities and relationships between them, and can use ontologies as a schema layer of reference. This way, it is possible to retrieve implicit knowledge through logical inference rather than only allowing queries that request explicit knowledge. Deep Learning methods have emerged from machine learning approaches and became essential for the resolution of several tasks within the artificial intelligence spectrum. Recently, Deep Learning methods have been used in conjunction with Knowledge Graphs (i.e., to represent relationship of the graph in a vector space, to allow companies find patterns in real-time between interconnected entities, to keep track of inventories of parts further allowing finding materials used in different products, etc.). Therefore, it has become critical that the Deep Learning and Knowledge Graphs communities join their forces in order to develop more effective algorithms and applications. This workshop aimed at reinforcing the relationships between these communities and intended to be at the center of shared works around topics such as Deep Learning, Knowledge Graphs, Natural Language Processing, Computational Linguistics, Big Data, and so on.

Therefore, the goal of this workshop was to provide a meeting forum where discussions between the relevant stakeholders (researchers from academia, industry and businessmen) could be stimulated within the Deep Learning and Knowledge Graphs domains. As the previous edition, this year we noticed a general attention to our workshop given the more than 10 submissions we received and the high number of participants we noticed during the workshop day. Eight papers have been accepted and discussed within the workshop by authors from different international institutions. They covered topics such as Knowledge Graph embeddings, entity summarization, entity type prediction, semantic entity enrichment. We had as invited speaker Prof. Aldo Gangemi who discussed how knowledge graph embeddings are both an opportunity and a matter of concern for the cognitive scientist, what patterns can be found and what else can be discovered in the direction of human-centred semantics. We really thank him for his great talk. We also thank the program committee for their time and work for reviewing the submitted papers. Although the workshop was held remotely due to the COVID-19 pandemic, it has been successful by more than 60 participants from all around the world. On the workshop website[6] it is possible to see screenshots reflecting some moments of the workshop.

December 2021

Mehwish Alam
Davide Buscaldi
Michael Cochez
Francesco Osborne
Diego Reforgiato Recupero
Harald Sack

[6] https://alammehwish.github.io/dl4kg2021/

Contents

Quality Assessment of Knowledge Graph Hierarchies using KG-BERT

Kinga Szarkowska[1], Véronique Moore[2], Pierre-Yves Vandenbussche[2] and Paul Groth[1]

[1]*University of Amsterdam, Science Park 904, 1098 XH Amsterdam, Netherlands*
[2]*Elsevier Amsterdam, Radarweg 29a, 1043 NX Amsterdam, Netherlands*

Abstract
Knowledge graphs in both public and corporate settings need to keep pace with the constantly growing amount of data being generated. It is, therefore, crucial to have automated solutions for assessing the quality of Knowledge Graphs, as manual curation quickly reaches its limits. This research proposes the use of KG-BERT for a triple (binary) classification task that assesses the quality of a Knowledge Graphs's hierarchical structure. The use of KG-BERT allows the textual as well structural aspects of a Knowledge Graph to be leverage for this quality assessment (QA) task. The performance of our proposed approach is measured using four different Knowledge Graphs: two branches (Physics and Mathematics) of a corporate Knowledge Graph - OmniScience, a WordNet subset, and the UMLS Semantic Network. Our method yields high-performance scores on all four KGs (88-92% accuracy) making it a relevant tool for quality assessment and knowledge graph maintenance.

Keywords
Ontology Maintenance, Knowledge Graphs, Hierarchical Knowledge Graphs, hierarchy evaluation, triple classification, contextual word embeddings, BERT, KG-BERT.

1. Introduction

Knowledge Graphs are at the heart of many data management systems nowadays, with applications ranging from knowledge-based information retrieval systems to topic recommendation [1]and have been adopted by many companies [1]. Our research originated with the need for the automatic quality assessment (QA) of OmniScience [2], Elsevier's cross-domain Knowledge Graph powering applications such as the Science Direct Topic Pages.[1]

A Knowledge Graph (KG) is a graph representation of knowledge, with entities, edges, and attributes [3]. Entities represent concepts, classes or things from the real world, edges represent the relationships between entities, and attributes define property-values for the entities. We refer to these sets as "triples".

A number of QA dimensions have been identified for KGs [4]. Here, we focus on the semantic accuracy dimension: the degree to which data values correctly represent the real-world facts (or concepts) [4]. More specifically, we focus on the hierarchy evaluation of KGs: whether

ISWC 2021: Deep Learning for Knowledge Graphs, October 24–28, 2021, Virtual Conference
kinga.szarkowska@gmail.com (K. Szarkowska); v.malaise@elsevier.com (V. Moore); p.vandenbussche@elsevier.com (P. Vandenbussche); p.t.groth@uva.nl (P. Groth)

[1]https://www.sciencedirect.com/topics/index

its hierarchical structure is correctly represented. To do this, we employ contextual word embeddings [5] and investigate the use KG-BERT method [6]'s binary classification task. It is a binary triple classification task, as for a given KG triple (entity, relation, entity/literal) the classifier will return whether a given triple is correct or not.

KG-BERT takes advantage of the textual representation of the data for assessing the veracity of KG triples and uses transfer learning to fine-tune embeddings pre-trained using the BERT model [5] for a triple classification task. *Our novel contribution is to use textual representations of the KG hierarchy in combination with KG-BERT to evaluate hierarchy quality.* We evaluate the performance of this method on four different KGs.

The paper is structured as follows: in the related work section, we present KGs evaluation frameworks and approaches for triple classification tasks. In the methodology section, we describe the datasets that were used for this research, along with our sampling strategy, a detailed description of the KG-BERT method and the evaluation metrics that we have selected. In the results section, we present results for the four selected KGs. Lastly, we summarize the outcomes of our research and propose future directions for exploration.

2. Related Work

In this section, we present KGs quality assurance frameworks and methods developed for KGs triple classification tasks.

2.1. Knowledge Graph Quality Assurance Frameworks

One of the first frameworks that has been widely used for the data quality assessment of a hierarchical triples structure was developed in 1996 by Wang and Strong [7]. They identified four main quality dimensions: intrinsic, contextual, representational, and accessibility [7]. A framework proposed in 2007 [8] and built on [7] introduces a taxonomy of Information Quality dimensions. Zaveri et al. [4] focused on a quality evaluation framework for linked data, gathering 18 quality dimensions (with 69 quality metrics), including the dimensions introduced by [7]. Chen et al. [9] adjusted the framework proposed by [4] for KGs. They created 18 requirements on knowledge graphs quality and mapped them to knowledge graph applications. Raad and Cruz [10] also gathered evaluation methodologies for ontologies. Raad and Cruz distinguished several validation criteria (such as accuracy, completeness, consistency) and presented four evaluation approaches.

The dimension that maps to the evaluation (assessment) of automatically expanding large scale KGs is the semantic accuracy dimension [4], or just accuracy [10]. Zaveri et al. [4] define it as the degree to which data values correctly represent real-world concepts. Our research develops a method to evaluate one component of this dimension: whether the hierarchy of a KG is accurate.

2.2. Knowledge Graph Triple Classification

We propose to use a (binary) triple classification task approach to address the problem of KG hierarchy evaluation. We describe how our method compares with the current state-of-the-art in KG evaluation.

A triple classification task aims to predict whether an unseen Knowledge Graph triple is true or not. For that we can consider KGs entities and their relationships as real-valued vectors, and assess the triples plausibility. Real-valued vector representations are called Knowledge Graph embeddings [11]. There are two types of approaches for creating the KG embeddings: translational distance models and semantic matching models [11]. The main difference is the use of scoring function: translational models use distance-based functions and semantic matching models use similarity-based scoring functions. Examples of translational models are: TransE [12], TransH [13], TransR [14], or TransG [15]. RESCAL [16] and DistMult [17] are representatives of semantic matching models. All of these methods use only the data's structural information, and do not leverage external information such as entity type, or textual description, to improve model performance.

KG-BERT was introduced to address this limitation by taking advantage of textual information in addition to structural information [6]: instead of creating Knowledge Graph embeddings to represent the structure of the data (using the relationships presented in the data), it uses a textual (distributed) representation of the triples in a corpus. It first creates a representations of each of the entities in a triple using pre-trained BERT [5] embeddings. These embeddings are then updated to minimize the loss function for the triple classification task. Finally, a binary classifier predicts from its embedding, whether a triple is true or not with an associated confidence score.

One consequence of initialization through BERT is that the model takes advantage of knowledge encoded within BERT during its training. This feature was one of the reasons to explore the use of KG-BERT for the this evaluation task.

3. Methodology

We elaborate now on our technical and methodological decisions. First, we present the datasets that are used in this research. Each dataset is from a different domain and used in a different community, but they all hierarchically structured. We introduce our negative sampling approaches as we evaluate our results against a gold set of triples that should only be true. We then describe KG-BERT in more detail, along with the evaluation approach.

3.1. Datasets

We selected four Knowledge Graphs from different domains (described in more details in Table 1), which are all hierarchical knowledge graphs (structured with the classification inclusion [18] relation). We discarded here the benchmark datasets of WN11 and FB13 typically used for triple classification tasks, as they contain a wide variety of relationships and were not representative of our target KGs.

WordNet [19] is a large lexical database representing relationships between words in the English language. It is widely used for tasks such as natural language processing or image classification [20]. We extracted a subset of WordNet focused on the classification inclusion

Table 1
Overview of the experimental datasets

Dataset	Example of the Triple	No. of Triples	No. of Entities
WordNet	*Refried Beans* is a *Dish*	62,323	48,048
UMLS	*Enzyme* is a *Chemical*	500	135
Physics	*Seismology* is a concept of *Geophysics*	5,404	3,697
Mathematics	*Outlier* is a concept of *Summary Statistics*	2,770	2,126

relation: we extracted only nouns - excluding proper names - in a hyponymy relation. We performed this filtering on the hyponymy Wordnet subset.[2]

The Unified Medical Language System (UMLS) [21] is a comprehensive ontology of biomedical concepts. UMLS is made of two distinct KGs: the Semantic Network and a Metathesaurus of biomedical vocabularies. The Semantic Network represents broad subject categories together with their semantic relations. It groups and organizes the concepts from the Metathesaurus in one of the four relationships: "consist of", "is a", "method of", "part of".

We selected the Semantic Network for our research because it has more generic concepts than the Metathesaurus and is better structured. We decided to investigate the performance of our model on the subset of that network consisting of the triples in a hierarchical ("is a") relationship (the classification inclusion relation).

OmniScience [2] is an all-science domains knowledge graph that is used, amongst others, for the annotations of research articles. It connects external publicly available vocabularies with the entities required in Elsevier platforms and tools. It is maintained by scientific experts from Elsevier. OmniScience is a poly-hierarchy, in which scientific concepts can belong to multiple domains. The relationship between OmniScience concepts can be described as a hyponymy relation, or "is a" relation. OmniScience has several domain branches, such as Physics, Mathematics, or Medicine and Dentistry. We used two branches as test cases, namely Physics and Mathematics.

3.2. Negative Sampling

We considered all of the KGs above to consist of correct triples. Therefore, negative sampling is necessary to prepare a training set for a classification problem. We followed the approach proposed in [22] and use the 1:3 ratio for negative samples. We followed three strategies to generate these negative samples:

1. per each head entity we randomly sample a tail entity;
2. per each tail entity we randomly sample a head entity;
3. per each pair we exchange the head entity with the tail entity, which gives us “reversed” samples, that should help train the model with respect to the direction of the relation.

After sampling 3 negative examples per each proper pair, we filtered out all of the generated samples did that occur in the original set of triples from KG to ensure that there are not contradictory samples in the training set.

[2]The subset is available at https://www.w3.org/TR/wordnet-rdf/.

3.3. Data preparation

Two approaches of dataset preparation were selected. In the first approach, negative examples were generated globally and then the dataset was split into three subsets: training, validation, and test dataset. A proportion 80/10/10 was used for WordNet and OmniScience branches. For the UMLS split 90/5/5 was selected, as the dataset is much smaller and we wanted to give the model a higher number of training examples.

In the second approach, in order to test the ability of the model to generalize to unseen triples, the KGs triples were first divided into three datasets (with the same proportion as before). Then for each of the datasets, negative triples were generated separately. In the end, we excluded the examples that occurred in the intersection of the datasets. A summary of the datasets used for the experiments is presented in Table 4.

Table 2
Number (#) of correct and incorrect triples

Experiment	Performance Experiment			Generalization Experiment		
Dataset	Train Set	Val Set	Test Set	Train Set	Val Set	Test Set
WordNet	49,075 / 99,481	6,096 / 12,474	6,070 / 12,499	48,797 / 148,294	6,210 / 18,659	6,221 / 18,670
UMLS	447/957	28/51	25/53	362/840	47/134	50/127
Physics Branch	4,340 / 12,857	565 / 1,585	499 / 1,651	4,321 / 12,835	541 / 1,613	540 / 1,609
Mathematics Branch	2,221 / 6,536	272/823	277/818	2,215 / 6,561	277/822	277/825

3.4. KG-BERT

KG-BERT is a state-of-the-art method [6] for triple classification tasks. The main idea behind it is to represent the KG triples as text, using their labels to create a lexicalization in natural language and gather contextual sentences from a corpus. This text can then be used to fine-tune the existing pre-trained BERT embeddings, for a classification task. As part of this lexicalization, we explored a set of equivalent ways to represent the notion of hyponymy. In our cases, where we had access to a large corpus *is a concept of* gave the best performance for the model for OmniScience and *is a* for UMLS.

The format of the model's input is as follows: each of the triple elements is separated by the [SEP] token, and at the beginning of the input a [CLS] token is added. Each entity name, as well as the relation's textual description is tokenized. An example of such a representation for the text *"Linear Algebra is a concept of Mathematics"* is: *[CLS] linear algebra [SEP] is a concept of [SEP] mathematics [SEP].* In the case of the absence of words in the vocabulary, the model is considering their sub-words and is adding specified tokens for the missing parts of the sentence. Cross-entropy is used as a loss function. For our research, we used the code[3] provided by the authors [6] as a basis. We obtained good performance with "bert-base-uncased" BERT base.

[3]https://github.com/yao8839836/kg-bert

4. Experiments and Results

In this section, the performance of the KG triple classification task and its ability to generalize are discussed. By Pp and Pn we refer to the value of precision for positive classes and negative classes respectively, by Rp and Rn we refer to the value of recall for positive and negative classes respectively. Our primary goal is to use the method for KG maintenance, therefore precision for positive examples and recall for negative examples is important (rather than a combined F1 value). With a high score of precision for positive examples, we want to be sure that all returned positives are true positives. With recall for negative examples, we want to be sure that all of the negative examples will be returned by the method, meaning every potential incorrect triple will be returned. Therefore we will focus on these two metrics.

Table 3 presents performance scores for the experiments. WordNet and UMLS were trained using *is a* as relation phrase, and OmniScience branches were trained with *is a concept of*. All of the models were trained using random seed equal to 42, and *bert-base-uncased* as a BERT base. Below we comment in detail on the results.

Table 3
Performance of the model

Dataset	Accuracy	Precision: P/N		Recall: P/N		TP	FN	FP	TN
WordNet	90.33%	84.65%	93.16%	86.03%	92.42%	5,222	848	947	11,552
UMLS	91.02%	80.00%	97.92%	96.00%	88.68%	24	1	6	47
Physics	92.41%	81.23%	96.15%	87.58%	93.88%	437	62	101	1,550
Mathematics	88.12%	75.43%	92.68%	78.70%	91.32%	218	59	71	747
Results generalization:									
WordNet	90.22%	81.82%	92.86%	78.27%	94.20%	4,869	1,352	1,082	17,588
UMLS	81.35%	60.49%	98.96%	98.00%	74.80%	49	1	32	95
Physics	91.53%	80.07%	95.53%	87.04%	93.04%	470	70	112	1,497
Mathematics	91.01%	87.08%	92.11%	75.45%	96.24%	209	68	31	794
Results domain generalization:									
Physics	79.77%	55.1%	89.95%	69.34%	82.92%	346	153	282	1,369
Mathematics	76.98%	52.91%	92.49%	81.95%	75.31%	227	50	202	616

4.1. Performance KG-BERT for different hierarchical Knowledge Graphs

The KG-BERT method applied as a triple classifier is tested using dataset splits described in Table 4. First, the best set of hyperparameters is selected. For this, the model was tested with a combination of: different numbers of epochs, of learning rates, of maximum sentences length, and of training batch size using a grid search approach. We chose the model with the highest accuracy score on the validation set.

We note a high (>88%) accuracy for all of the KGs, and also high scores for all the metrics selected by us as important (Pp and Rn). OmniScience's Physics Branch evaluation gets the highest scores, while the Mathematics Branch gets the lowest for that experiment.

4.2. Model generalization to fully unseen triples

We investigated the model's ability for generalizing on unknown data. We performed two experiments: the first of them (generalization), applied to all four datasets, uses the second split of data reported in Table 4, and the second sampling approach described in subsection 3.3; the second experiment (domain generalization) tested specifically whether a model trained on one OmniScience branch could be applied to another branch. We tested the model trained on one branch with the test set generated for the other branch, for both branches.

Again, we note a high (>90%) accuracy in the generalization experiments for all of the KGs, except from UMLS dataset, where the accuracy is equal to 80%. We discuss the performance on UMLS further in the Discussion section. The Physics Branch classification result gets the highest accuracy score, but the Mathematics Branch achieved the highest scores in independent values for Pp and Rn.

The results for the domain generalization experiment that tested how a model trained on the Mathematics branch of OmniScience performed when classifying examples from the Physics branch (and another way round) are as follows: 80% accuracy for Physics branch being a test set, and 77% for Mathematics branch. Pp is equal to 55% and 53% accordingly, and Rn is equal to 83% and 75%. Even if the accuracy score and Rn are relatively high, we note the low ability of the model to properly classify positive examples (Pp equal to 53-55% in both cases). This means that a model should be trained per domain for a better performance.

4.3. Prediction of long-distance hierarchy using the model

We carried out an experiment to check whether the model can predict a hierarchical relationship between concepts further apart in the hierarchical structure. We created some datasets with triples containing concepts at different levels or hierarchical depth (different hop-levels). As a point of terminology, given the two triples *Optics is a concept of Physics* and *Fiber Optics is a concept of Optics*, we consider the triple *Fiber Optics is a concept of Physics* a 2-hop triple. The three datasets are described in table 4 .

Table 4
Number (#) of correct and incorrect triples for long-distance hierarchy prediction experiment

Dataset	Train Set	Val Set	Test Set	
			1-hop	2-hop
WordNet	49,075 / 99,481	6,096 / 12,474	6,070 / 12,499	96,236 / 288,246
Omni. Physics Branch	4,340 / 12,857	565 / 1,585	499 / 1,651	3,532 / 10,325
Omni. Math Branch	2,221 / 6,536	272 / 823	277 / 818	1,099 / 3,054

For all of the datasets, scores such as accuracy, Pp, and Rn are decreasing with the increase of hop distance. For Pp the decrease is substantial in every dataset (20-30% decrease). For WordNet Pp decreased to 51% for the 2-hop triples, for Physics to 68%, and for Mathematics Branch Pp decreased to 45%. Rn decreased less suddenly (5-8%). For 3-hop, 4-hop, and pairs with more concepts in-between Pp and Rn continued to decrease. This shows that our model performs well to predict a direct hypernym relationship, but not for hypernym at more than one hop distance.

4.4. Evaluation of classification error output

We performed an error analysis on the incorrectly classified triples output. We found triples such as *blue is a clothing*, *cold is an apartment* or *smoker is a passenger car* were classified as a FP examples in WordNet: these are clearly pure errors of the model's output. In the FP examples for Mathematics and Physics branch of OmniScience, we noticed words overlap between two concepts from the considered triples. For Mathematics branch 43.5% of FP have 3-letter overlapping subwords, almost 30% FP have 4-letter, 23% have 5-letter, and 17.4% have 6-letter overlap. For Physics branch we noted 3-letter overlap for 36.63% of FP, 28% FP noted 4-letter, 21.8% have a 5-letter, and almost 20% have 6-letter overlap. Therefore, we can see that the model has difficulties with establishing the hierarchy between concepts with a large lexical overlap in their naming.

5. Discussion

Performance scores across the selected metrics for QA are high (Precision for positive triples and Recall for negative triples). For every dataset, we noted an accuracy score above 88%. Accuracy for the generalization experiment, where we wanted to see how the model can deal with examples that it did not use for training, yielded decent results (accuracy above 80%). However, testing a model build on one OmniScience branch on another scientific domain of the same KG did not give accurate results. Therefore, the model can generalize well, but needs to be trained per domain.

The reproducibility of the method tested here is a strong point: we showed that this approach can be used on four very different datasets used in different research contexts. However, the accuracy for the UMLS dataset was not as high as the others. The main reason for the lack of performance is the small size of the sample. We further investigated the converge of the model, and concluded that for OmniScience's Physics and Mathematics datasets, the model started to learn with around 1.7k examples. Moreover, for the Physics branch we noted good results from 6.8k training examples (40% of the data), and for Mathematics branch from around 5.6k (65 % of the data). These proportions should be taken into consideration when applying our method.

Our recommendation on how to prepare the model for hierarchy evaluation is as follows:

- the model should be trained per domain,
- relation sentence (or lack of it) could be selected as one of the hyperparameters, and used for the models' optimization,
- proper negative sampling strategy should be selected, the one that can be the most representative strategy of the possible errors in the KGs,
- depending on the scientific domain, different BERT bases could be selected (e.g. *bio-clinical-bert* [23] or *sci-bert* [24] could be used for KGs in the medical or biological domain).

6. Conclusion

In this study, we explored the use of contextual word embeddings for quality assessment of knowledge graph hierarchies. We used KG-BERT, which uses textual information about KGs triples to enrich structure-based embeddings. We described how to use this approach for quality assessment and showed that it works well for both large scale corporate knowledge graphs as well as subsets of one publicly available knowledge graph (WordNet). Moreover, we tested whether the model can generalize across unseen examples and between domains.

We see different paths for future work. First, exploring how to apply the method for the KG maintenance in a real-life practical settings: besides using this framework for a global QA pipeline in real life, this method can also be tested to propose a candidate placement in an existing KG for an incoming candidate concept (by assessing the plausibility of all possible combinations of that candidate with existing concept from a given domain or KG). We have observed good empirical results, but still have to test the idea at scale for the automatic development of KGs.

Secondly, testing the method on other KGs, particularly KGs that have a less consistent hierarchical structure would add value to our understanding of the limitations of the use of BERT embeddings for quality assessment.

In terms of QA methods for the triples assessment, a gold set based approach (assessing the quality of a KG by rebuilding it and comparing it with the original set) could help assessing the feasibility of this method for the automatic generation of large KGs.

References

[1] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor, Industry-scale knowledge graphs: Lessons and challenges, Commun. ACM 62 (2019) 36–43. URL: https://doi.org/10.1145/3331166. doi:10.1145/3331166.

[2] V. Malaisé, A. Otten, P. Coupet, Omniscience and extensions – lessons learned from designing a multi-domain, multi-use case knowledge representation system, in: C. Faron Zucker, C. Ghidini, A. Napoli, Y. Toussaint (Eds.), Knowledge Engineering and Knowledge Management, Springer International Publishing, Cham, 2018, pp. 228–242.

[3] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, 2021. arXiv:2003.02320.

[4] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey, Semantic Web 7 (2015) 63–93. doi:10.3233/SW-150175.

[5] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[6] L. Yao, C. Mao, Y. Luo, KG-BERT: BERT for knowledge graph completion, CoRR abs/1909.03193 (2019). URL: http://arxiv.org/abs/1909.03193. arXiv:1909.03193.

[7] R. Wang, D. Strong, Beyond accuracy: What data quality means to data consumers, J. Manag. Inf. Syst. 12 (1996) 5–33.

[8] B. Stvilia, L. Gasser, M. Twidale, L. Smith, A framework for information quality assessment, Journal of the Association for Information Science and Technology 58 (2007) 1720–1733. doi:10.1002/asi.20652.

[9] H. Chen, G. Cao, J. Chen, J. Ding, A Practical Framework for Evaluating the Quality of Knowledge Graph, 2019, pp. 111–122. doi:10.1007/978-981-15-1956-7_10.

[10] J. Raad, C. Cruz, A survey on ontology evaluation methods, 2015. doi:10.5220/0005591001790186.

[11] Y. Dai, S. Wang, N. Xiong, W. Guo, A survey on knowledge graph embedding: Approaches, applications and benchmarks, Electronics 9 (2020) 750. doi:10.3390/electronics9050750.

[12] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NIPS, 2013.

[13] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: AAAI, 2014.

[14] H. Lin, Y. Liu, W. Wang, Y. Yue, Z. Lin, Learning entity and relation embeddings for knowledge resolution, Procedia Computer Science 108 (2017) 345–354. URL: https://www.sciencedirect.com/science/article/pii/S1877050917305628. doi:https://doi.org/10.1016/j.procs.2017.05.045, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[15] H. Xiao, M. Huang, X. Zhu, Transg : A generative model for knowledge graph embedding, 2016, pp. 2316–2325. doi:10.18653/v1/P16-1219.

[16] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data., 2011, pp. 809–816.

[17] B. Yang, W.-t. Yih, X. He, J. Gao, l. Deng, Embedding entities and relations for learning and inference in knowledge bases (2014).

[18] J. J. Odell, Advanced object-oriented analysis and design using UML, Cambridge ; New York : Cambridge University Press ; New York : SIGS Books, 1998. URL: http://www.loc.gov/catdir/toc/cam024/97018368.html, includes bibliographical references and index.

[19] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, Language, Speech, and Communication, MIT Press, Cambridge, MA, 1998.

[20] A. Benitez, S. Chang, Image classification using multimedia knowledge networks, Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429) 3 (2003) III–613.

[21] O. Bodenreider, The unified medical language system (umls): Integrating biomedical terminology, 2004.

[22] B. Athiwaratkun, A. Wilson, Hierarchical density order embeddings, ArXiv abs/1804.09843 (2018).

[23] E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Naumann, M. McDermott, Publicly available clinical BERT embeddings, in: Proceedings of the 2nd Clinical Natural Language Processing Workshop, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 72–78. URL: https://www.aclweb.org/anthology/W19-1909. doi:10.18653/v1/W19-1909.

[24] I. Beltagy, K. Lo, A. Cohan, Scibert: A pretrained language model for scientific text, in: EMNLP, 2019.

Language Models As or For Knowledge Bases

Simon Razniewski[a], Andrew Yates[a,b], Nora Kassner[c] and Gerhard Weikum[a]

[a]*Max Planck Institute for Informatics*
[b]*University of Amsterdam*
[b]*LMU Munich*

Abstract
Pre-trained language models (LMs) have recently gained attention for their potential as an alternative to (or proxy for) explicit knowledge bases (KBs). In this position paper, we examine this hypothesis, identify strengths and limitations of both LMs and KBs, and discuss the complementary nature of the two paradigms. In particular, we offer qualitative arguments that latent LMs are not suitable *as* a substitute for explicit KBs, but could play a major role *for* augmenting and curating KBs.

1. Introduction

The ability of pre-trained contextual language models (LMs) to capture and retrieve factual knowledge has recently stirred discussion as to what extent LMs could be an alternative to, or at least a proxy for, explicit knowledge bases (KBs). LMs, such as BERT [1], GPT [2] or T5 [3] are huge transformer-based neural networks trained in a self-supervised manner on huge text corpora, in order to predict sentence completions or masked-out text parts. In a setting called (masked) prompting or probing [4], these LMs complete a text sequence intended to elicit a relational assertion for a given subject. For example, GPT-3 correctly completes the phrase *"Alan Turing was born in"* with *"London"*, which can be seen as yielding a subject-predicate-object triple ⟨ `Alan Turing, born in, London` ⟩.

Starting from the LAMA probe [5], many works have explored whether this LM-as-KB paradigm could provide an alternative to structured knowledge bases such as Wikidata. Exemplary analyses investigated the inclusion of entity information [6], how to turn LMs into structured KBs [7], and how to incrementally add knowledge without side effects [8]. Other work studied how accuracy relates to the neural network's storage capacity [9] and whether QA performance scales with model size [10]. Another focus area is how LMs-as-KBs can be further augmented with a text retrieval component, to include informative passages (e.g., from Wikipedia) [11, 12, 13].

Although most works make their speculative nature clear (e.g., the title of [5] ends with a question mark), there is an implicit suggestion that LMs could replace structured KBs. On the other hand, NLP-centric works have identified various kinds of inconsistencies in LM outputs [14] or questioned their quantitative performance [15].

This paper discusses the potential of LMs *as* KBs and its "softer" variation of LMs *for* KBs.

Deep Learning for Knowledge Graphs (DL4KG)
srazniew@mpi-inf.mpg.de (S. Razniewski); ayates@mpi-inf.mpg.de (A. Yates); kassner@cis.lmu.de (N. Kassner); weikum@mpi-inf.mpg.de (G. Weikum)

	LM-as-KB	Structured KB
Construction	Self/Unsupervised 👍	Manual or semi-automatic 👎
Schema	Open-ended 👍	Typically fixed 👎
Maintenance		
- adding facts	Difficult, unpredictable side effects 👎	Easy 👍
- correcting/deleting	Difficult 👎	Easy 👍
Knows what it knows	No, assigns probability to everything 👎	Yes, content enumerable 👍
Entity disambiguation	No/limited 👎	Common 👍
Provenance	No 👎	Common 👍

Table 1
Differences of LMs-as-KBs and structured KBs

2. Background

LM-as-KB refers to efforts to use an LM as a source of world knowledge, as proposed by [5]. The knowledge representation is inherently latent, given by the entirety of the neural network's parameter values (in the billions). LMs in general have greatly advanced tasks like text classification, machine translation, information retrieval, and question answering (see, e.g., survey [16]).

KBs, on the other hand, have been steadily advanced since the mid 2000s (with early works like DBpedia, Freebase and Yago) [17]. They represent knowledge in the form of subject-predicate-object (SPO) triples along with qualifiers for non-binary statements. KBs have become key assets in major industry applications [18, 19], including search engines. A major issue for ongoing KB research is quality assurance as the KB is grown and maintained. This includes human-in-the-loop approaches throughout the KB life-cycle [20, 21, 22].

All LM-as-KB examples that follow are based on the GPT-3 daVinci model [2], one of the largest pre-trained LMs as of October 2021.

3. LM-as-KB

3.1. Intrinsic Considerations

The following are principal differences between LMs-as-KBs and structured KBs.

Predictions vs. lookups: While content of structured KBs can be explicitly looked up, LMs have a latent representation and output probabilities at probing time. This has the advantage of not requiring any schema design upfront. However, it implies that it is not possible to enumerate the knowledge stored in an LM, nor can we look up whether a certain fact is contained or not. For predictions with very high confidence scores, this is still viable. However, even top-ranked predictions often have low scores and near-ties. Properly calibrating scores and thresholding is a black art.

Example: GPT-3 does not have tangible knowledge that Alan Turing was born in London; it merely assigns this a high confidence of 83%. Yann LeCun, on the other hand, is given medium confidence in being a citizen of France and Canada (67% and 26%), but he actually has French and USA citizenship, not Canadian. The LM assigns USA a very low score. The Wikidata KB, on the *other hand, only states his French citizenship, not USA. Wikidata is incomplete, but it does not contain any errors.*

Statistical correlations vs. explicit knowledge: Errors made by LMs-as-KBs are not random, but exhibit systematic biases [23, 15] due to frequent values and co-occurrences (including indirect co-occurrences captured latently).

Example: When prompting GPT-3 for awards won by Alan Turing, its top-confidence prediction is the Turing Award, and lower-ranked outputs include "Nobel Prize" and "the war" (none of them correct).

Awareness of limits: In KBs, absence of facts is explicit and easy to assert. Wikidata even supports a way of stating non-existence (no-value statements) to impose a local-closed-world view while following a general open-world assumption [24]. LM's latent representations inherently lack awareness of cases where no object exist, and so they easily produce non-zero or even high scores for incorrect assertions.

Example: Alan Turing was homosexual and never married. When prompting GPT-3 with the phrase "Alan Turing married", the top prediction is "Sara Lavington" with score 21%, and for the prompt "Alan Turing and his wife" it is "Sara Turing" (his mother's name). This is a case of LM hallucination [25, 26]. In contrast, Wikidata has an explicit statement ⟨ `Alan Turing, spouse, no value` ⟩ *denoting that he was unmarried.*

Coverage: The scope of KBs is usually limited by the fixed set of predicates specified in the KB schema. These can be hundreds (or even a few thousands) of interesting relations, but will hardly ever be complete. In particular, "non-standard relations", such as `worked with colleague`, `song is about person` (or `event`), `movie based on person's biography`, are missing in all of the major KBs. LMs, on the other hand, latently tap into the full text of Wikipedia, books, news, and more, and are thus able to capture some of these predicates.

Example: Creatively prompting GPT-3 can yield impressive nuggets of knowledge: the input phrases "Turing's colleague" and "Turing worked with" result in outputs like John Womersley, Hugh Alexander, Gordon Welchman (all correct). Likewise, the prompt "The Imitation Game film is about the life of" is completed with the high-confidence output Alan Turing. These anecdotes indicate the great power of LMs to go beyond the current scope and coverage of explicit KBs.

Curatability: In structured KBs, a knowledge curator can correct, add or remove assertions. For LMs, this is an open challenge, as these operations require major (non-monotonic) re-training, or the addition of explicit exceptions, which means reverting to a KB [27, 28].

Example: For the prompt "Alan Turing died in the town of", GPT-3 returns the top prediction "Warrington", which is wrong (he died in Wilmslow). The LM does not provide any hint on how to fix this (e.g., by changing the training corpus or parameters), and a knowledge curator has no way to tackle such errors.

Provenance: LMs have no ability to trace their outputs back to specific source documents (and passages) in the training data. KBs, on the other hand, consider reference sources as an indispensable pillar of scrutable veracity. Provenance is crucial for giving explanations to users, including knowledge engineers who maintain the KB and end-users in downstream applications. Also, without provenance, LMs have no way of pinpointing an incorrect prediction's root cause and correcting the underlying corpus (e.g., removing misleading documents).

Example: Reconsider the previous example of predicting "Warrington" as Turing's death place. The LM itself does not give any cue where this comes from. A diligent and smart Google user could detect a possible origin, namely, news and other reports about a memorial plaque at 2 Warrington Crescent in Maida Vale, London, which is near Turing's birth place. However, the knowledge engineer cannot be certain that this is indeed the culprit.
Correctly predicted facts need explanations, too. For example, the assertion that Turing was engaged with Joan Clarke may appear puzzling given his homosexuality. Pointing to explicit provenance is crucial evidence.

3.2. Pragmatic Considerations

Entity disambiguation: Although LMs are lauded for their ability to disambiguate words based on context, this happens latently, and there is no easy way to explicitly build this into probing procedures [9, 6]. Consequently, LMs mix up facts from distinct entities that share surface forms. Although structured KBs cannot perform disambiguation on their own either, they can correctly separate assertions.

Example: GPT-3 completes "Turing was a famous" with "mathematician", "computer", "code" etc., stemming from very different entities (including the Turing Machine).

Numbers and singletons: LMs are good at latently capturing knowledge about predicates with few possible object values, such as nationality or language-spoken. However, when the object values are rarely occurring values or even singletons (i.e., occurring only with a single subject), the latent representation is bound to produce errors, and explicit KB storage is superior. The same applies to many cases of numeric values, where the value distribution exhibits high entropy.

Example: For the input "The Turing Institute's address in London is", GPT-3 returns "Dilly's Den" or "the street called Dilly's Den" (possibly derived from the famous Piccadilly Circus; the correct value is British Library, 96 Euston Road, London NW1 2DB). Rephrasing the prompt does not lead to success either.

Subjects with zero or many objects: An important case where the brittleness of LM predictions becomes a significant problem is when a subject entity has no object value for a given predicate or has many distinct true values. The zero-value case often leads to the pitfall that the LM must predict some value. In the many-values case, we could go deep in the ranking of the LM output, but this would usually result in a wild mix of valid and spurious objects, and there is no guideline for how deep we should go into the ranking.

Example: To obtain a list of Turing Award winners, we could prompt GPT-3 with the phrase "the Turing Award was won by" and receive various predictions like "Stuart Shieber", "John Hopcroft" and "Andrew Yao" (1 false, 2 correct). There are currently 73 winners, all captured in Wikidata. By probing LMs, we would have to go very deep in the prediction ranking to see all of them, but only in a confusing mix of true and false positives.
As for zero-objects, the prompt for "the first woman on the moon was" returns Sally Ride, Eileen Collins and others. These are astronauts, but unfortunately, none of them ever landed on the moon. The ground-truth for this example is empty.

We summarize the main differences in Table 2.

4. LM-for-KB

Our view of how to harness the great potential of LMs is to leverage them *for KB curation*: maintaining high quality as the KB grows throughout its life-cycle. This is a major pain point in KB practice [20, 21, 22]. For example, when adding new entities, one needs to ensure that they are not duplicates (with slightly different alias names) of existing entities. Likewise, keeping the type system (aka ontology) clean while gradually extending it and ensuring the correctness of new facts are never-ending challenges.

The envisioned role of LMs is to *scrutinize SPO assertions* considered for augmenting the KB. For example, a new fact such as ⟨ `Leonardo da Vinci, has won, Turing Award` ⟩ could be "double-checked" by prompting the LM as to whether it yields high-confidence predictions for this candidate assertion. This is akin to the way knowledge graph embeddings [29] have been considered for KB completion. However, the key difference is that KG embeddings draw from the KG itself, and thus do not provide complementary evidence. LMs, on the other hand, bring in a new and largely independent perspective, by tapping into text corpora (including Wikipedia, but also news, books etc.). If the LM does not yield sufficiently confident support for the candidate fact, it should be refuted.

The converse direction, using LMs to predict assertions and thus generate candidates for new facts, is conceivable too. However, this needs major research to advance prediction accuracy.

5. Conclusion

In this paper we discussed the strengths and limitations of LMs *as* KBs in comparison to structured KBs. We believe that LMs cannot broadly replace KBs as explicit repositories of structured knowledge. While the probabilistic nature of LM-based predictions is suitable for task-specific end-to-end learning, the inherent uncertainty of outputs does not meet the quality standards of KBs. LMs cannot separate facts from correlations, and this entails major impediments for KB maintenance. We advocate, on the other hand, that LMs can be valuable assets *for* KB curation, by providing a "second opinion" on new fact candidates or, in the absence of corroborated evidence, signal that the candidate should be refuted. Other ways of combining the strengths of latent knowledge (LMs) and structured knowledge (KBs) could be promising as well, such as "KB-for-LM" approaches that allow a LM to look up facts from an external memory (e.g., [12, 30, 31, 32]) and thus have the potential to combine the strengths of both approaches.

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL, 2019.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019. OpenAI technical report.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, arXiv (2019).

[4] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, arXiv (2021).

[5] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, A. Miller, Language models as knowledge bases?, in: EMNLP, 2019.

[6] N. Poerner, U. Waltinger, H. Schütze, E-BERT: Efficient-yet-effective entity embeddings for BERT, in: Findings of EMNLP, 2020.

[7] C. Wang, X. Liu, D. Song, Language models are open knowledge graphs, arXiv (2021).

[8] R. Wang, et al., K-adapter: Infusing knowledge into pre-trained models with adapters, arXiv (2021).

[9] B. Heinzerling, K. Inui, Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries, in: EACL, 2021.

[10] A. Roberts, C. Raffel, N. Shazeer, How much knowledge can you pack into the parameters of a language model?, in: EMNLP, 2020.

[11] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, S. Riedel, How context affects language models' factual predictions, in: AKBC, 2020.

[12] K. Guu, K. Lee, Z. Tung, P. Pasupat, M.-W. Chang, Realm: Retrieval-augmented language model pre-training, in: ICML, 2020.

[13] P. Lewis, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, in: NeurIPS, 2021.

[14] Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, Y. Goldberg, Measuring and improving consistency in pretrained language models, arXiv (2021).

[15] B. Cao, H. Lin, X. Han, L. Sun, L. Yan, M. Liao, T. Xue, J. Xu, Knowledgeable or educated guess? revisiting language models as knowledge bases, in: ACL, 2021.

[16] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, Computational intelligence magazine (2018).

[17] S. Razniewski, P. Das, Structured knowledge: Have we made progress? an extrinsic study of KB coverage over 19 years, in: CIKM, 2020.

[18] N. Noy, et al., Industry-scale knowledge graphs: lessons and challenges, CACM (2019).

[19] G. Weikum, L. Dong, S. Razniewski, F. Suchanek, Machine knowledge: Creation and curation of comprehensive knowledge bases, in: Foundations and Trends in Databases, 2021.

[20] J. Taylor, ~~Automated~~ knowledge base construction, AKBC invited talk, 2020. https://youtu.be/JsB4T35We0w?t=12032.

[21] A. Piscopo, E. Simperl, What we talk about when we talk about Wikidata quality: a literature survey, in: Symposium on Open Collaboration, 2019.

[22] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe, P. Szekely, A study of the quality of Wikidata, arXiv (2021).
[23] E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, On the dangers of stochastic parrots: Can language models be too big?, in: FAccT, 2021.
[24] H. Arnaout, S. Razniewski, G. Weikum, J. Z. Pan, Negative knowledge for open-world Wikidata, in: Companion Proceedings of the Web Conference, 2021.
[25] A. Rohrbach, et al., Object hallucination in image captioning, in: EMNLP, 2018.
[26] C. Wang, R. Sennrich, On exposure bias, hallucination and domain shift in neural machine translation, in: ACL, 2020.
[27] C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. Yu, S. Kumar, Modifying memories in transformer models, in: arXiv, 2020.
[28] N. D. Cao, W. Aziz, I. Titov, Editing factual knowledge in language models, in: EMNLP, 2021.
[29] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, TKDE (2017).
[30] T. Févry, L. B. Soares, N. FitzGerald, E. Choi, T. Kwiatkowski, Entities as experts: Sparse memory access with entity supervision, in: EMNLP, 2020.
[31] H. Sun, L. B. Soares, P. Verga, W. W. Cohen, Adaptable and interpretable neural memory over symbolic knowledge, in: NAACL, 2021.
[32] N. Kassner, O. Tafjord, H. Schutze, P. Clark, Enriching a model's notion of belief using a persistent memory, in: arXiv, 2021.

GraphPOPE: Retaining Structural Graph Information Using Position-aware Node Embeddings

Jeroen B. den Boef[1,2], Joran Cornelisse[2] and Paul Groth[1]

[1]University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands
[2]Socialdatabase, Slego 1A, 1046 BM Amsterdam, The Netherlands

Abstract
Exponential computational cost arises when graph convolutions are performed on large graphs such as knowledge graphs. This computational bottleneck, dubbed the 'neighbor explosion' problem, has been overcome through application of graph sampling strategies. Graph Convolutional Network architectures that employ such a strategy, e.g. GraphSAGE, GraphSAINT, circumvent this bottleneck by sampling subgraphs. This approach improves scalability and speed at the cost of information loss of the overall graph topology. To improve topological information retention and utilization in graph sampling frameworks, we introduce Graph Position-aware Preprocessed Embeddings (GraphPOPE), a novel, feature-enhancing preprocessing technique. GraphPOPE samples influential anchor nodes in the graph based on centrality measures and subsequently generates normalized geodesic, Cosine or Euclidean distance embeddings for all nodes with respect to these anchor nodes. Structural graph information is retained during sampling as the position-aware node embeddings act as a skeleton for the graph. Our algorithm outperforms GraphSAGE on a Flickr benchmark dataset. Moreover, we demonstrate the added value of topological information to Graph Neural Networks.

Keywords
Graph Convolutional Networks, Graph Neural Networks, Graph Topology, Feature Embeddings

1. Introduction

Data that emphasizes relationships between data points, e.g. social networks, general knowledge, protein interactions, can be formally represented as graphs. Within machine learning there has been much interest in leveraging these graphs representations leading to the inception of graph learning and Graph Neural Networks (GNN) [1]. Graph neural networks have been successfully applied to a wide variety of tasks utilizing graph-structured data ranging from knowledge graphs to social networks [2, 3, 1, 4].

Many of the initial teething problems of GNNs have been resolved [5, 6, 7, 8, 9, 10]. Graph Convolutional Networks (GCN) combined a convolutional smoothing kernel with a spectral graph representation to achieve state of the art results on transductive node classification tasks [5]. While accurate in a transductive setting, this approach to node classification tasks fails to generalize well to unseen nodes [6]. GraphSAGE opened up the avenue for inductive graph

ISWC2021: Workshop on Deep Learning for Knowledge Graphs (DL4KG), October 25, 2021
jeroen@socialdatabase.com (J. B. d. Boef); joran@socialdatabase.com (J. Cornelisse); p.t.groth@uva.nl (P. Groth)
https://pgroth.com/ (P. Groth)
0000-0001-5649-2778 (J. B. d. Boef); 0000-0003-0183-6910 (P. Groth)

convolution models by learning general embedding generation functions for node features [6]. The GraphSAGE propagation rule utilizing a mean aggregator function is nearly equivalent to the one utilized in transductive GCNs and can be viewed as a linear approximation of localized spectral convolution. Additionally, primitive skip connections are performed by concatenating previous neighborhood representations of nodes with the current neighborhood representation. GraphSAGE also introduced a complementary NeighborSampler dataloader which improved scalability by introducing mini-batch training to Graph Convolutional Networks. The NeighborSampler aids embedding computation by sampling neighboring nodes iteratively and constructing mini-batches of nodes. Bipartite graphs are subsequently constructed to simulate the computation flow of GNNs. While inductive by nature and competitively accurate when introduced, the GraphSAGE model is constrained by its set neighborhood sampling function, as this restricts the convolutional kernel to a fixed size. The introduction of Graph Attention Networks (GAT) resolved this constraint by using an attention mechanism as a dynamic smoothing kernel [7]. When combined with a self-attention mechanism, this approach to graph convolution produces competitive results on both inductive and transductive tasks [7, 11].

1.1. Present work

Graph sampling architectures improve scalability and speed for Graph Convolutional Networks on large graphs at the cost of information loss with respect to overall graph topology. In an effort to improve topological information retention and utilization in graph sampling frameworks, we propose a general preprocessing technique for Neural Networks operating on graph-structured data, called GraphPOPE (Graph POsition-aware Preprocessed Embeddings). In this framework, topological information is embedded into the feature matrix through the generation of relative distance embeddings. By sampling *anchor nodes* from a given graph, identification points are determined. Normalized relative distance embeddings are then generated for all pairings of nodes and anchor nodes. These embeddings serve as a skeleton of the graph and identify which neighborhood a node belongs to. Intuitively, GraphPOPE embeddings can be interpreted as node2vec neighborhood embeddings for the whole graph, whereas node2vec generates second-order random walks neighborhood embeddings for individual nodes [12]. This makes GraphPOPE applicable to Multi Layer Perceptrons and local pooling models such as Graph Convolutional Networks alike, as topological information beyond the scope of a convolutional kernel is provided.

2. Related Literature

Conceptually, GraphPOPE is closely related to recent advances in GNNs that seek to improve topological information usage through position-aware convolutional layers. We consider GraphPOPE closely intertwined with graph sampling techniques as it mitigates topological information loss.

[0]Code implementations are publicly available on Github: https://github.com/JeroendenBoef/GraphPOPE

2.1. Graph Sampling Approaches

to GNNs favor scalability and speed over embedding or self-attention strategies by sampling subgraphs for training. Notable instances of this approach are Cluster-GCN and GraphSAINT, which both address the main computational bottleneck of GCNs [8, 9]. This computational bottleneck has been dubbed the *neighbor explosion* problem and is twofold: First, outputs of a GCN for a single node require data from neighbouring nodes in the previous layer of the network [8, 9]. Every layer within a GCN requires another n-hop neighbors where n depends on the convolutional kernel size, increasing computational cost exponentially for every layer. Second, back-propagation of a GCN requires all of the embeddings in the computation graph to be stored in GPU memory. Cluster-GCN proposes a solution to this bottleneck by preceding training with a clustering phase. Clusters of nodes belonging to dense subgraphs are identified during this clustering phase. These subgraphs are then used to restrict neighborhood search, acting as boundaries for the convolutional kernel. This relatively simple strategy introduces scalability to graph convolutional networks while reducing computational costs by a large margin. However, the employed clustering algorithm introduces additional heavy computational costs.

GraphSAINT adopts a similar approach to Cluster-GCN, marginally improving accuracy but substantially decreasing computation time [9]. The improved computational speed is mainly achieved through employment of inexpensive sampling algorithms, contrasting the expensive clustering algorithm utilized by Cluster-GCN.

2.2. Information Loss:

While sampling graphs during training mitigates the neighbor explosion problem and reintroduces scalability to GCNs, it nevertheless results in the loss of information. Restricting the GCN to specific clusters or completely disconnected subgraphs during training withholds or even removes edges and thus information from the graph. Chiang and colleagues identify this shortcoming for Cluster-GCN and introduce stochastic multiple clustering in an effort to reduce clustering bias and restore lost information simultaneously [8]. However, stochastic multiple clustering exclusively addresses the issue of cut edges during stochastic gradient descent batch updates, disregarding information loss preceding this phase.

A residual weakness within GNNs is their inability to distinguish between node positions with regards to the broader context of graph structure [13]. Not taking node features into account, two nodes can reside within opposite sides of a graph while having a topologically identical neighbourhood structure. Attempted heuristics range from attempts at deeper GNNs to node feature augmentation using position-aware convolutional layers [10].

3. Method: GraphPOPE

The inclusion of position-aware node embeddings as a general preprocessing technique for graph-based Neural Networks is motivated by the perceived topological information loss in graph sampling GCNs. Embedding this information into the node features before subgraph sampling could improve model performance. We first describe the geodesic GraphPOPE algorithm, which samples anchor nodes stochastically and subsequently generates position-aware

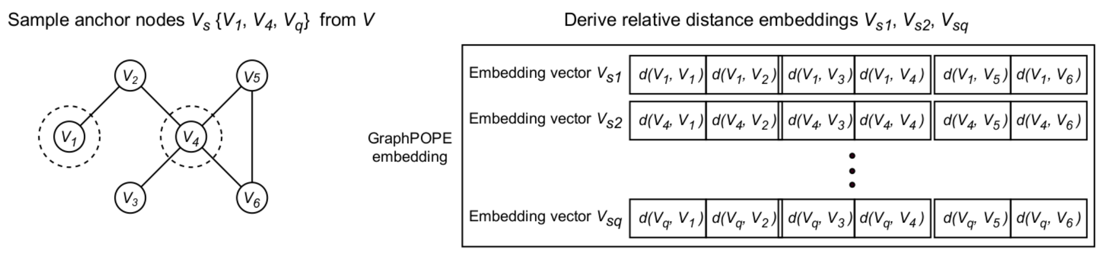

Sample anchor nodes V_s {V_1, V_4, V_q} from V
V_1
V_2
V_3
V_4
$V5$
V_6
Derive relative distance embeddings V_{s1}, V_{s2}, V_{sq}
GraphPOPE embedding
Embedding vector V_{s1}
$d(V_1, V_1)$ $d(V_1, V_2)$ $d(V_1, V_3)$ $d(V_1, V_4)$ $d(V_1, V_5)$ $d(V_1, V_6)$
Embedding vector V_{s2}
$d(V_4, V_1)$ $d(V_4, V_2)$ $d(V_4, V_3)$ $d(V_4, V_4)$ $d(V_4, V_5)$ $d(V_4, V_6)$
Embedding vector V_{sq}
$d(V_q, V_1)$ $d(V_q, V_2)$ $d(V_q, V_3)$ $d(V_q, V_4)$ $d(V_q, V_5)$ $d(V_q, V_6)$

Figure 1: Schematic overview of the GraphPOPE embedding generation

node embeddings for all nodes in a given graph (Section 3.1). Embedding enhancement through biased anchor node sampling and algorithmic time complexity are detailed in Section 3.2. Finally, we introduce a faster, embedding space approximation of the geodesic GraphPOPE in Section 3.3. A schematic overview of the GraphPOPE algorithm is depicted in figure 1.

3.1. Geodesic Distance Embeddings

This section details the GraphPOPE anchor node sampler and geodesic distance embedding generator algorithm (Algorithm 1 - GraphPOPE-geodesic), which assumes the output matrix is concatenated with the feature matrix so that all nodes are enriched with their respective distance embeddings. Let $\mathscr{G}(\mathscr{V}, \mathscr{E})$ denote graph $\mathscr{G}$ with nodes $\mathscr{V}$ and edges $\mathscr{E}$, n the amount of anchor nodes $\mathscr{V}_s$ to sample, d the geodesic distance function used to derive relative node distances and $D^{N \times n}$ the geodesic distance matrix generated by GraphPOPE. The intuition behind this algorithm is that for each node v_i in the graph, normalized geodesic distances between this node and all sampled anchor nodes $\mathscr{V}_s$ are computed and added to feature vector $\mathbf{v}_v$, which is subsequently added to the relative distance matrix D at index i. Anchor nodes are sampled stochastically to reduce algorithm complexity and prevent bias in the data. The distance function employed for this computation is either a single-source or all-pairs shortest path algorithm, returning 0 if the target node is unreachable and $\frac{1}{d(v_i,u_j)}$ otherwise. As this distance function serves as an approximation of how many hops a node is from an anchor node, it can be replaced by similar but faster distance functions.

3.2. Biased Anchor Node Sampling

The vanilla GraphPOPE (Algorithm 1) avoids bias through stochastic sampling of anchor nodes. This approach to sampling has a potential drawback of sampling less influential nodes. We introduce biased anchor node sampling based on node centrality which replaces the stochastic sampler in Algorithm 1 and alleviates this aforementioned phenomenon.

In this algorithm (see Appendix A, Algorithm 2: Biased Sampler), centrality scores are derived for all nodes and the highest ranking nodes are selected as anchor nodes. This extension upon the vanilla GraphPOPE algorithm aims to increase and stabilize the amount of topological

[1] j is utilized as an enumeration of $u \in \mathscr{V}_s$ in line 3 to insert d_{ij} into vector v_v at index j

Algorithm 1 GraphPOPE-geodesic

```
Input: Graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$; Sampling amount $n$; Distance function $d$
Output: Geodesic distance matrix $D^{N \times n}$
1: Stochastically sample $n$ anchor nodes $\mathscr{V}_s$ from $\mathscr{V}$
2: for $v_i \in \mathscr{V}$ do
3:     for $u_j \in \mathscr{V}_s$ do
4:         $d_{ij} \leftarrow \frac{1}{d(v_i,u_j)}$
5:         Embedding vector $\mathbf{v}_v[j] \leftarrow d_{ij}$ [1]
6:     end for
7:     $D[i] \leftarrow \mathbf{v}_v$
8: end for
```

information encoded in the distance embeddings by selecting nodes with higher centrality scores.

The geodesic distance matrix generation of Algorithm 1 employs a single-source shortest path algorithm for distance function d with time complexity $O(V + E)$ [14]. With this function being utilized for every combination of sampled anchor node V_s with every node $v_i \in V$, this results in an overall complexity of $O(V(V_s(V + E)))$. This can be simplified to a notation of $O(V_s(V^2 + VE))$, which is close to a worst-case complexity of $O(V^4)$ for a densely connected, directional graph ($E = V^2$) with $V_s = V$. This is nevertheless an extreme scenario, divergent from most average cases. When treated as an all-pairs shortest path problem on a directed graph and computed in parallel, this complexity can be improved to $O(V(V + E))$. An example of such an approach is the Floyd-Warshall algorithm, which generates a complete mapping of all shortest paths in a given graph. While this all-pairs approach to the shortest path derivation reduces the complexity from approximately $O(V^3)$ to $O(V^2)$, it is substantially more costly with regards to memory usage.

Biased anchor node sampling through node centrality introduces additional time complexity. Betweenness centrality would likely identify well connected, influential nodes most accurately as it denotes the fraction of shortest paths that pass through a given node. Nodes with a high betweenness centrality score would logically be more connected than those with a lower score, and thus less likely to return 0 when fed into distance function d. As this centrality measure requires shortest path computation, biased sampling has similar scalability issues to the shortest path algorithms employed in Algorithm 1. As faster approximations for betweenness centrality, other measures such as eigenvector-, clustering coefficient-, degree-, farness- and closeness centrality are utilized for centrality function c [15].

3.3. Embedding Space Approximation

In order to resolve the exponential scaling complexity of GraphPOPE-geodesic, we propose an embedding space alternative. This algorithm (see Appendix A, Algorithm 3: GraphPOPE-node2vec) utilizes Node2vec to generate local neighborhood embeddings $\mathscr{V}_E$ for every node v_i in $\mathscr{V}$ [12]. Anchor nodes can then be sampled stochastically or with a bias by K-means clustering $\mathscr{V}_E$ into n clusters and utilizing the cluster centroids as *pseudo anchor nodes* $\mathscr{V}_{Es}$. Classical

K-means has a time complexity of $O(n^2)$ which can be reduced to $O(n)$ through cluster shifting [16]. Moreover, the necessity of biased sampling is reduced for this algorithm, as edge traversal is not utilized in the distance computation. As a result, information loss does not occur in a similar fashion to the geodesic distance calculation. Distance between $\mathscr{V}_E$ and $\mathscr{V}_{Es}$ is then calculated through parallelized matrix multiplications, which has a linear complexity of $O(n)$. Algorithms used for the distance function are Cosine similarity, Cosine distance and Euclidean distance.

4. Experiments

We evaluate position-aware node embeddings by performing node property predictions on two benchmark datasets: Flickr and PubMed [9, 17]. In all experiments, predictions are performed on nodes that are unseen during training but included in the preliminary GraphPOPE embedding generation. This is thus considered a supervised, transductive learning setting. We used Weights & Biases for experiment tracking and hyperparameter optimization [18]. Sections 4.1 and 4.2 detail the experimental setup and data, respectively.

4.1. Experimental Setup

Experiments were conducted with an Ubuntu OS, GTX 2060 and RTX 3060 NVIDIA GPUs and Intel i7-5820K and Intel gold 6130 CPUs. Geodesic distances are derived with Networkx and all models are implemented through a combination of Pytorch, Pytorch Geometric and an abstraction layer of Pytorch Lightning [19, 20, 21, 22]. Our baseline model is a vanilla GraphSAGE architecture, consisting of 2-3 SAGE convolutional layers with intermediate batch normalization layers, the GraphSAGE mini-batch NeighborSampler, a hidden dimension size of 256, a dropout rate of 0.5 and a cross-entropy loss function [23, 6, 21]. GraphPOPE models used for experimentation are divided into two categories, geodesic and node2vec approaches. Geodesic iterations consist of the vanilla GraphPOPE-stochastic version and its biased alternatives. Specifically betweenness-, closeness-, degree-, clustering coefficient-, eigenvector centrality and PageRank-based versions. Node2vec implementations are normalized Cosine and Euclidean distances supported by biased anchor node sampling through K-means clustering. All experiments were conducted utilizing identical architecture and hyperparameters with the exception of optimized batch sizes, amount of convolutional layers (2 or 3) and GraphPOPE settings.

Hyperparameter optimization was conducted separately per dataset for GraphPOPE-geodesic, GraphPOPE-node2vec and the vanilla GraphSAGE baseline to ensure unbiased comparison. These optimizations are implemented through hyperparameter sweeps with a Bayesian search method to optimize validation accuracy [18]. Sweeps are performed over n anchor nodes, batch size B and the amount of convolutional layers ℓ.

Experiments are run for a max of 300 epochs using early stopping mechanics and a learning rate monitor. A starting learning rate of 0.001 is employed which is reduced upon stagnation of validation loss. Finally, an Adam optimizer is used for all models and early stopping is provoked if validation accuracy does not increase for 20 epochs [24, 25].

The motivation behind this experimental setup is twofold. It allows us to assess whether graph sampling GCNs can be improved through introduction of additional topological information. Moreover, it reduces the possibility of experimental bias as the convolutional kernel introduces unfavourable conditions. Convolutional kernels already have access to topological information of local node neighborhoods, rendering the information gain of GraphPOPE less potent.

4.2. Data

Two graph datasets of contrasting size and density are employed. Test partition nodes are unseen during training with a separate dataloader that is exclusively instantiated upon conclusion of training.

Flickr dataset We use the Flickr dataset introduced with GraphSAINT [9]. In the dataset, edges denote shared metadata among images such as locations or users. Labels are manually merged tags and represent 7 entities such as animals, nature, humans, etc. Features are 500-D bag of words extractions of SIFT image descriptions. The graph contains roughly 89,250 nodes with 899,756 edges connecting them. Similar to Zeng and colleagues, edge weights are normalized in-degrees and a fixed-partition split is applied to the data, resulting in 50/25/25 train/val/test split [9].

Citation dataset The PubMed medical citation graph is used to assess performance on a smaller, less challenging node property prediction dataset [17]. In this directed graph, nodes represent scientific publications regarding diabetes research from the PubMed database, edges indicate outgoing citations and labels represent publication categories. The dataset consists of 19,717 nodes divided over 3 classes with 44,338 edges. We employ the FastGCN partition split, resulting in 500 validation and 1000 test nodes, leaving the remaining 18,217 nodes for training [26].

5. Results

We provide experimental results detailing accuracy metrics on the tasks, feature importance of *n* anchor nodes and hyperparameter sweeps. Reported accuracy scores are averages of 20 runs in a range of fixed global seeds to ensure reproducibility. Tables 1 detail the results. Accuracy scores are accompanied by their respective standard error and highest accuracy values are denoted in bold. on benchmarking tasks. Optimized hyperparameters are given in Appendix A, Table 2.

Table 1 shows an accuracy increment for all geodesic GraphPOPE models with respect to the baseline on the Flickr dataset. Moreover, GraphPOPE-geodesic implementations that employ biased sampling of anchor nodes generally experience more substantial accuracy gains. Contrastingly, node2vec-based approximations yield no improved performance. Results on PubMed indicate a homogeneous performance of 89% accuracy.

Results on Flickr indicate that configurations with more anchor nodes generally experience a more substantial increase in performance except for node2vec approximations. Increasing the amount of anchor nodes from 32 to 256 for an unoptimized, geodesic GraphPOPE with closeness centrality sampling raises accuracy from 51.98% to 53.05%. Moreover, validation accuracy yields an 88% positive correlation with the amount of anchor nodes over 90 hyperparameter sweeps on the aforementioned configuration. PubMed experiments display a contrasting trend where the amount of anchor nodes provide diminishing returns. On this smaller dataset, the amount of anchor nodes have a linear correlation of -25%, resulting in an optimal configuration with 64 anchor nodes to maximize validation accuracy.

Table 1
Averaged prediction results over 20 runs on optimized hyperparameters. Biased K-means anchor node sampling is utilized for N2V implementations.

	Flickr	PubMed
Name	Acc	Acc
Betweenness centrality	52.93 ± 0.23	89.14 ± 0.67
Closeness centrality	52.55 ± 0.24	89.28 ± 0.65
Degree centrality	52.92 ± 0.23	89.32 ± 0.63
Clustering coefficient	52.63 ± 0.23	89.40 ± 0.59
Eigenvector centrality	52.48 ± 0.17	89.29 ± 0.60
PageRank	**52.94** ± 0.25	89.05 ± 0.55
Stochastic	52.75 ± 0.24	**89.55** ± 0.56
N2V-cdist	51.70 ± 0.29	89.52 ± 0.39
N2V-Euclidean	51.68 ± 0.30	89.52 ± 0.39
Baseline	51.78 ± 0.17	89.51 ± 0.35

6. Discussion

Our experimental results on the Flickr dataset display trends of accuracy gain for GraphPOPE-enhanced architectures with a more substantial improvement on larger datasets. Additionally, accuracy gains improve upon biased anchor node sampling. The amount of anchor nodes correlates positively with an increase in validation accuracy. This suggests that additional topological information can be beneficial to sampling-assisted Graph Convolutional Networks. Specifically for larger graphs, where the fraction of topological information beyond the convolutional kernel is higher.

GraphPOPE-node2vec poses an ineffective embedding-space alternative. Whereas computation complexity is improved, model accuracy is not. This phenomenon might be explained by the fact that convolutional kernels employed by GCNs already have access to the local neighborhood information encoded by node2vec.

Scalability is a recurring bottleneck for GNNs. Our time complexity stems from the algorithm's geodesic distance calculation. Overcoming these scalability issues could prove beneficial for graph learning on large datasets, given the substantial accuracy increments on such graphs. Embedding-space approximations of the distance calculation could provide a solution for the memory and computation time bottlenecks, removing the need for costly edge traversal operations. Alternatively, models could be trained to approximate the distance function. Finally, deep learning alternatives for the identification of influential nodes in the graph could accelerate performance of biased anchor node sampling, potentially improving another time complexity component.

7. Conclusion

We introduced GraphPOPE, a novel prepossessing technique designed to improve topological information retention and utilization in Graph Neural Networks. Our algorithm is applicable to any Neural Network that has access to graph-structured data and operates through position-aware node embedding generation. We demonstrate an accuracy gain on graph benchmarking datasets Flickr and PubMed with the application of our position-aware node embeddings, which can be improved additionally at the cost of additional time complexity. Our experimental results indicate that larger graphs benefit more from increased amounts of topological information retention. Finally, we propose approximations for future research to reduce time complexity, thus increasing applicability to real-world scenarios.

References

[1] W. L. Hamilton, Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine Learning 14 (2020) 1–159.

[2] W. Fan, Y. He, Y. Ma, E. Zhao, D. Yin, Q. Li, J. Tang, Graph neural networks for social recommendation, arXiv (2019) 417–426.

[3] S. Arora, A survey on graph neural networks for knowledge graph completion (2020). arXiv:2007.12374.

[4] T. Thanapalasingam, L. van Berkel, P. Bloem, P. Groth, Relational graph convolutional networks: A closer look, CoRR abs/2107.10015 (2021). arXiv:2107.10015.

[5] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[6] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in Neural Information Processing Systems 2017-Decem (2017) 1025–1035. arXiv:1706.02216.

[7] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, arXiv (2017) 1–12. arXiv:1710.10903.

[8] W. L. Chiang, Y. Li, X. Liu, S. Bengio, S. Si, C. J. Hsieh, Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019) 257–266. doi:10.1145/3292500.3330925. arXiv:1905.07953.

[9] H. Zeng, H. Zhou, A. Srivastava, V. Prasanna, R. Kannan, GraphSAINT: Graph sampling based inductive learning method, arXiv (2019). arXiv:1907.04931.

[10] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, 36th International Conference on Machine Learning, ICML 2019 2019-June (2019) 12372–12381. arXiv:1906.04817.

[11] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, arXiv preprint arXiv:2005.00687 (2020).

[12] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[13] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826 (2018).
[14] A. Bhargava, Grokking Algorithms: An illustrated guide for programmers and other curious people, Simon and Schuster, 2016.
[15] X. He, N. Meghanathan, Alternatives to betweenness centrality: A measure of correlation coefficient, 2016. doi:10.5121/csit.2016.61301.
[16] M. Pakhira, A linear time-complexity k-means algorithm using cluster shifting, 2014. doi:10.1109/CICN.2014.220.
[17] Z. Yang, W. W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, 2016. arXiv:1603.08861.
[18] L. Biewald, Experiment tracking with weights and biases, 2020. URL: https://www.wandb.com/, software available from wandb.com.
[19] A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using NetworkX, Technical Report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library (2019) 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[21] M. Fey, J. E. Lenssen, Fast graph representation learning with pytorch geometric, arXiv preprint arXiv:1903.02428 (2019).
[22] W. Falcon, et al., Pytorch lightning, GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning 3 (2019).
[23] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167. arXiv:1502.03167.
[24] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[25] I. Loshchilov, F. Hutter, Fixing weight decay regularization in adam, CoRR abs/1711.05101 (2017). URL: http://arxiv.org/abs/1711.05101. arXiv:1711.05101.
[26] J. Chen, T. Ma, C. Xiao, Fastgcn: Fast learning with graph convolutional networks via importance sampling, CoRR abs/1801.10247 (2018). URL: http://arxiv.org/abs/1801.10247. arXiv:1801.10247.

A. Appendix

Algorithm 2 Biased sampler

```
Input: Graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$; Sampling amount $n$; Centrality function $c$
Output: Anchor nodes $\mathscr{V}_s$
    $C \leftarrow c(\mathscr{G})$
2:  Sample $n$ highest $C_i$ anchor nodes $\mathscr{V}_s$ from $\mathscr{V}$
```

Algorithm 3 GraphPOPE-node2vec

```
Input: Graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$; Sampling amount $n$; Distance function $d$; Node2vec algorithm $z$; K-
    means clustering algorithm $k$; Bias setting $b \in \{True, False\}$
Output: Normalized embedding distance matrix $\tilde{D}^{N \times n}$
    Node2vec embedding $\mathscr{V}_E \leftarrow Z(\mathscr{G})$
    if $b == True$ then
3:      clustering centroids $C \leftarrow K(\mathscr{V}_E)$
        Sample $n$ pseudo anchor nodes $\mathscr{V}_{Es}$ from $C$
    else
6:      if $b == False$ then
            Stochastically sample $n$ anchor nodes $\mathscr{V}_{Es}$ from $\mathscr{V}_E$
        end if
9:  end if
    $D \leftarrow d(\mathscr{V}_E, \mathscr{V}_{Es})$
    Normalize $D$
```

Table 2
Optimized hyperparameter settings for GraphSAGE and GraphPOPE-enhanced GraphSAGE. Hyperparameters depicted are n anchor nodes $\mathscr{V}_s$, batch size B and convolutional layers ℓ.

	Flickr			PubMed		
Name	$n\mathscr{V}_s$	B	ℓ	$n\mathscr{V}_s$	B	ℓ
GraphPOPE-geodesic	256	1550	3	64	800	2
GraphPOPE-node2vec	64	625	3	256	750	3
Baseline	-	3550	2	-	2200	2

Challenges of Applying Knowledge Graph and their Embeddings to a Real-world Use-case

Rick Petzold[2,3,4], Genet Asefa Gesese[1,3,5], Viktoria Bogdanova[2,4], Thorsten Zylowski[2,4], Harald Sack[1,3,5] and Mehwish Alam[1,3,5]

[1]*FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany*
[2]*CAS Software AG, Germany*
[3]*Karlsruhe Institute of Technology, Institute AIFB, Germany*
[4]*firstname.lastname@cas.de*
[5]*firstname.lastname@fiz-karlsruhe.de*

Abstract

Different Knowledge Graph Embedding (KGE) models have been proposed so far which are trained on some specific KG completion tasks such as link prediction and evaluated on datasets which are mainly created for such purpose. Mostly, the embeddings learnt on link prediction tasks are not applied for downstream tasks in real-world use-cases such as data available in different companies/organizations. In this paper, the challenges with enriching a KG which is generated from a real-world relational database (RDB) about companies, with information from external sources such as Wikidata and learning representations for the KG are presented. Moreover, a comparative analysis is presented between the KGEs and various text embeddings on some downstream clustering tasks. The results of experiments indicate that in use-cases like the one used in this paper, where the KG is highly skewed, it is beneficial to use text embeddings or language models instead of KGEs.

Keywords

Knowledge Graph Embedding, Language Models, Clustering

1. Introduction

As discussed in [1], according to the 2017 Kaggle Machine Learning & Data Science Survey the majority of data scientists use relational data in their work. In significant number of industries such relational data are modeled and stored in relational databases such as MySQL and Oracle. Data scientists make use of the data stored in these databases to perform different machine learning applications such as clustering and classification. However, in order to apply such algorithms directly to the data significant feature engineering efforts are required. Hence, one way to address this issue is to convert the relational databases into a Knowledge Graph (KG)

Woodstock'21: Symposium on the irreproducible science, June 07–11, 2021, Woodstock, NY

kulyabov-ds@rudn.ru (R. Petzold); Manfred.Jeusfeld@acm.org (V. Bogdanova); Manfred.Jeusfeld@acm.org (T. Zylowski)
http://conceptbase.sourceforge.net/mjf/ (V. Bogdanova); http://conceptbase.sourceforge.net/mjf/ (T. Zylowski)
0000-0001-7116-9338 (G. A. Gesese); 0000-0002-9421-8566 (V. Bogdanova); 0000-0002-9421-8566 (T. Zylowski); 0000-0002-9421-8566 (H. Sack); 0000-0002-9421-8566 (M. Alam)

and then learn embeddings for the obtained KG which in turn will be used as inputs to the downstream tasks.

The relational database (RDB) used in this paper is hosted by the company CAS Software AG[1] and it contains information about German companies, i.e., their addresses, contact persons, industrial sectors, and so on. The database is converted to a KG using the D2RQ [2] tool. In order to apply machine learning algorithms on the KG, it is necessary to transform the KG into low-dimensional vector space while preserving the semantics present in the KG. There exist various approaches proposed for such purposes like DistMult [3] and ComplEx [4]. However, if the created KG is highly skewed with not enough semantics present which is the exact scenario in our use-case, challenges arise when trying to learn representations for the KG, i.e., KGE models do not perform well on KGs with such characteristics. Experiments with some KGE models are conducted to prove this.

Another alternative to KGEs, is to leverage the textual descriptions of the companies and apply text-based embedding models to get latent representations for the companies. The textual descriptions of the companies are extracted from their respective websites. Some downstream company clustering tasks are performed using the representations learned using both the KGE models. The results of the clustering indicates the effectiveness of the text-embeddings over the KGEs. ExCut [5] performs clustering of entities by combining KG embeddings with rule mining methods. Even though ExCut also uses a real-world KG, the quality of the KG is better and suitable for applying KGEs as compared to the use-case (i.e., CAS-KG which is the KG generated from the RDB provided by CAS) that is being addressed in our paper. The contributions of this work are i) analysing real-world datasets for KG embeddings, ii) applying KG embeddings for a downstream task, and iii) comparing text and KG embeddings on real-world datasets.

The rest of the paper is organized as follows: Section 2 discusses the process of converting the RDB to KG followed by the challenges in mapping the KG to Wikidata and learning latent representations for the KG using KGEs. In Section 3, latent representation learning of companies using text embedding models is discussed. The experimental results on downstream clustering tasks are provided in Section 4 followed by the closing remarks in Section 5.

2. Generating KG from Relational Database

Here, converting the RDB to a KG is discussed along with the challenges that occur while trying to enrich the KG with external information and learn latent representations.

2.1. Applying D2RQ to Convert the Relational Database to a Knowledge Graph

The first step is cleaning the database by normalizing it to BCNF and filtering out unnecessary tables, i.e., tables with data that do not provide any useful information to learn representations for companies. After normalizing the database, it is converted to RDF in N-Triples format using D2RQ. As the result of the conversion, there are 5 entity types, 9,794,528 entities, 3 object relations, 21 datatype properties, 74,220,549 triples among which 12,138,554 contain object

[1]https://www.cas.de/start.html

relations and the rest 62.081.995 are triples with datatype properties. The entity types are Company (8,945,631), City (150,377), State (16), Legal Form (45), and Person (6,98,459).

Note that there is no any direct connection between two entities of the same type. Due to this fact, the generated KG (i.e., CAS-KG) is highly skewed and is not rich in semantics. In order to increase the quality of CAS-KG, it is beneficial to enrich the graph with external information.

2.2. Challenges in Mapping CAS-KG to Wikidata

As discussed above CAS-KG is required to be enriched with information from external sources. One of such sources is Wikidata which is a publicly available Linked Oped Data. An attempt has been made to map the companies that are in CAS-KG to items in Wikidata. However, the following two challenges arise when dealing with the mapping **i)** Most of the companies in the CAS-KG are small local businesses which do not have corresponding items in Wikidata. This is observed while trying to perform simple string-based comparison of the names of the companies in CAS with the labels of items that are of type Organization/Business/Company in Wikidata. **ii)** It was possible to map the entities of type LegalForm, and City to Wikidata items. For instance, the Legal Form GmbH in Cas could be mapped to GmbH (Q460178) in Wikidata. However, mapping entities of such types do not actually bring much of usable semantic enrichment without being able to map Companies.

2.3. Applying KGEs on CAS-KG

Here, the challenge is proven by applying some KGE-based link prediction task on CAS-KG. Since CAS-KG is huge in terms of triples and the total number of entities, it is necessary to select a sub-graph for the experiments, which is referred to as CAS286K. CAS286K contains 285,808 entities, 3 relations, 382,964 structured triples, 306,371 training triples, 38,296 test triples, and 38,297 validation triples. The dataset is available at https://github.com/rickpetzold/CAS-Knowledge-Graph.

DistMult [3] and ComplEx [4] KGE models are used to learn representations for the dataset CAS286K. These two models are selected to show the differences that they have in handling asymmetric relations, i.e., unlike ComplEx, DistMult does not perform well with asymmetric relation and all the 3 relations that exist in the CAS286K are asymmetric. Note that the choice of the KGE model does not affect the purpose of these experiments which is to prove that the KG lacks the required quality to apply a KGE model on it. Note that two different ways of initialization are used with the ComplEx model, i.e., random initialization (ComplEx) and initialization with fastText [6] embeddings (ComplEx$_{init}$). The fastText Embeddings are generated by averaging embeddings of the labels and keywords associated with the corresponding entities.

The Stochastic Local Closed World Assumption (sLCWA) [7] training approach is used with model optimization hyperparameter ranges - Embedding dimension: {64,128,256}, Optimizers: {Adam, AdaGrad}, Regularizers: {None, L1, L2}, Weight for L1 and L2: [0.01,1.0), lr:[0.001,0.1), batch size: {128,256,512,1024}, Loss: {BCEL, MRL}, Number of negatives: {1,2, . . . ,30}, and Margin for MRL: {0.5,1.5, . . . , 9.5}. Number of trials: 10, epochs:100, early stopping with patience of 50 epochs evaluating every 10 epochs. For ComplEx, the optimizer, the loss, and the regularizer are fixed to Adam, BCEL, and L2 respectively so as to reduce computational cost. The optimal hyperparameter values for DistMult and ComplEx are embedding dimension: 128 & 100, Regularizer: L2 & L2, weight: 0.025 & 0.0228, Loss: MRL & BCEL, negative sampler: 6 & 61, optimizer: Adam & Adam, and Batch Size: 256 & 512. Detailed information about sLCWA and the aforementioned loss functions is available in [7].

The results obtained are MRR 0.000034, 0.2, and 0.0074 for DistMult, ComplEx, and ComplEx$_{init}$ respectively. The values of each of these evaluation metrics are too low mainly with DistMult due to some characteristics of the CAS286K dataset which already makes it hard to learn embeddings using KGE approaches. Firstly, the entities of type Company have no incoming relations, i.e., they never occur as tails in the KG which makes the graph highly skewed. This indicates that there exist no single direct connection between any two entities of type Company. Since ComplEx is better than DistMult in dealing with asymmetric relations and most of the relations are asymmetric in CAS286K, the MRR with ComplEx (0.2) is better than with Distmult (0.000034). Moreover, even though initializing ComplEx with FastText embeddings is better than DistMult, it is not better than the randomly initialized ComplEx model due to the fact that only less than 1% of the entities of type company have keywords. Pykeen[2] is used to undertake the experiments.

3. Text Embeddings

As it has already been discussed in Section 2.2 and 2.3, applying the KGE approaches in such highly skewed KG with very limited links between entities is not beneficial. Hence, it is better to apply text embedding models instead as it could be more feasible to find textual descriptions for the companies. Therefore, web crawling is performed to get the textual descriptions of companies and while doing so, those websites containing either very short or non-german text are removed.

In order to learn representations for companies using the crawled texts, different embedding models are used separately, i.e., pretrained fastText and GloVe [8] embeddings, Multilingual Bert [9] & Sentence BERT [10] with/without fine tuning, and Multilingual Universal Sentence Encoder (MUSE) [11]. BERT is fine-tuned on a multiclass-classification task with and without removing stopwords, i.e., BERT$^{nostoprem}_{finetuned}$ using 4049 companies for training & 1200 for validation on 24 classes/sectors and (BERT$_{finetuned}$) using 2552 companies for training & 800 for validation on 16 classes/sectors.

4. Downstream task: Clustering

The clustering task is to group together companies based on their industrial sectors. A gold standard dataset is created with 503 companies where the maximum, minimum, average number of tokens in the textual descriptions of these companies are 695, 209, and 547.67. This gold standard contains 12 industry sectors (classes) in total, the sectors and their corresponding number of companies are: 'Photographers (86)', 'Onlineshop (51)', 'Webdesigner (51)', 'Coaching, Training, and Workshop (50)', 'Real Estate Agent (50)', 'Dentist (50)', 'Advertising Agencies (46)',

[2]https://pykeen.readthedocs.io/en/stable/

Table 1
Clustering results using text, KG, and combined embeddings

		HDBSCAN		BIRCH		K-Means	
	Model	AMI	silh.	AMI	silh.	AMI	silh.
Text embeddings	fastText	0.187	-0.082	0.205	0.056	0.198	0.08
	GloVe	0.234	0.093	0.582	0.132	0.543	0.13
	BERT	0.245	0.017	0.463	0.125	0.432	0.1
	BERT$_{finetuned}$	0.324	-0.022	0.467	0.229	0.46	0.256
	BERT$^{nostoprem}_{finetuned}$	0.488	**0.16**	0.547	0.263	0.548	0.314
	SentenceBERT	0.456	0.068	0.614	0.147	0.627	0.15
	SentenceBERT$_{finetuned}$	0.195	0.143	0.237	**0.451**	0.216	**0.523**
	MUSE	**0.579**	0.158	**0.692**	0.221	**0.676**	0.206
KG embeddings	DistMult	0.013	-0.087	0.01	0.057	0.003	0.106
	ComplEx	0.006	-0.022	0.003	-0.094	0.0003	-0.128
	ComplEx$_{init}$	0.014	-0.053	0.012	0.088	0.007	0.004
Combined	MUSE + ComplEx$_{init}$	0.015	-0.03	0.01	0.074	0.006	0.151

'Consulting (37)', 'IT Services (25)', 'Online Agencies (23)', 'Attorney (21)', and 'Travel Agencies (13)'.

BIRCH [12], HDBSCAN [13], and K-means are selected for the clustering task. For BIRCH the hyperparameters are the number of clusters 1-20, branching factor 10-200, and threshold 0.1-0.9. For HDBSCAN the minimal samples are 1-50 and the minimal cluster size is 2-100 whereas for K-means the number of clusters is 2-30. As the result in Table 1 indicates, the text embeddings give better results as compared to the KG embeddings in the clustering task. This is due to the highly skewed nature of the CAS286K dataset. Information from external resources is required in order to improve the KGE results. Note that, the MRR results with ComplEx$_{init}$ is not better than ComplEx on the link prediction task. However, the opposite holds on the clustering task because 82% of the companies in the gold standard have keywords. Note that the combined embeddings are generated by simply concatenating representations from MUSE and ComplEx$_{init}$.

5. Conclusion

In this paper, the challenges in applying KGE models on real world use-cases are discussed. Experiments on clustering tasks are conducted using as inputs latent representations learned by applying both KGEs and text embeddings separately. The results of the experiments prove the initial analysis that are made about KGEs not working well on datasets with very low quality such as CAS286K.

References

[1] M. Cvitkovic, Supervised learning on relational databases with graph neural networks, arXiv preprint arXiv:2002.02046 (2020).

[2] C. Bizer, A. Seaborne, D2rq-treating non-rdf databases as virtual rdf graphs, in: Proceedings of the 3rd International Semantic Web Conference, 2004.

[3] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: International Conference on Learning Representations (ICLR), 2015.

[4] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, ICML'16, JMLR.org, 2016, p. 2071–2080.

[5] M. H. Gad-Elrab, D. Stepanova, T. Tran, H. Adel, G. Weikum, Excut: Explainable embedding-based clustering over knowledge graphs, in: Proceedings of 19th International Semantic Web Conference, 2020, pp. 218–237.

[6] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2016).

[7] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, J. Lehmann, Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework, arXiv preprint arXiv:2006.13365 (2020).

[8] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[9] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: NAACL, 2019.

[10] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019.

[11] D. M. Cer, Y. Yang, S. yi Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, R. Kurzweil, Universal sentence encoder, ArXiv abs/1803.11175 (2018).

[12] T. Zhang, R. Ramakrishnan, M. Livny, Birch: An efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.

[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, AAAI Press, 1996, p. 226–231.

Knowledge Graph Embeddings or Bias Graph Embeddings? A Study of Bias in Link Prediction Models

Andrea Rossi[a], Donatella Firmani[b] and Paolo Merialdo[a]

[a]*Roma Tre University, Roma, Italy*
[b]*Sapienza University, Roma, Italy*

Abstract
Link Prediction aims at tackling Knowledge Graph incompleteness by inferring new facts based on the existing, already known ones. Nowadays most Link Prediction systems rely on Machine Learning and Deep Learning approaches; this results in inherent opaque models in which assessing the robustness to data biases is not trivial. We define 3 specific types of Sample Selection Bias and estimate their presence in the 5 best-established Link Prediction datasets. We then verify how these biases affect the behaviour of 9 systems representative for every major family of Link Prediction models. We find that these models do indeed learn and incorporate each of the presented biases, with a heavily negative effect on their behaviour. We thus advocate for the creation of novel more robust datasets and of more effective evaluation practices.

Keywords
Link Prediction, Knowledge Graph Embeddings

1. Introduction

Knowledge Graphs (KGs) are structured repositories of information where nodes modeling real-world *entities* are linked by labeled directed edges; each label represents a semantic *relation*, therefore each edge linking a pair nodes represents a *fact* conveying that the corresponding two entities are connected via that relation.

KGs have recently achieved widespread popularity in a variety of contexts. Large open KGs, such as DBpedia [1], YAGO [2] and Wikidata [3], are used on a daily basis for Semantic Web projects [4] and question answering [5]. Meanwhile, a growing number of companies rely on private KGs to support their services. Google and Microsoft use respectively the Google KG [6] and Satori [7] to enhance their search engines; Amazon [8] and Ebay [9] use product graphs to improve their recommendations; social networks like Facebook [10] and LinkedIn [11] use KGs for user profiling and advertisement.

It is therefore unsurprising that many KGs have achieved web-scale dimensions, featuring millions of entities and billions of facts. Nonetheless, it is well-known that even even the largest

DL4KG 2021: Workshop on Deep Learning for Knowledge Graphs, held as part of ISWC 2021: the 20th International Semantic Web Conference, October 24 - 28, 2021 (Online Event)
andrea.rossi3@uniroma3.it (A. Rossi); donatella.firmani@uniroma1.it (D. Firmani); paolo.merialdo@uniroma3.it (P. Merialdo)

and richest KGs suffer from *incompleteness*, as they only hold a small portion of the real-world knowledge they should encompass [12].

Link Prediction (LP) tackles this issue by leveraging the already known facts in the KG to infer new ones. Nowadays the vast majority of LP models learn vectorized representations of entities and relations called *embeddings* using Machine Learning (ML) techniques; many of them rely on Deep Learning architectures, featuring sequences of neural layers interspersed by activation functions. These models have been shown to achieve state-of-the-art performances [13, 14]. In the last decade embedding-based LP has become a sparkling research area, with dozens of novel models being proposed every year (see the work by Wang *et al.* [15] for a comprehensive survey). The pioneering model TransE [16] has established the practice of evaluating these systems computing global metrics of their predictive performances over datasets obtained from real-world KGs.

LP datasets are usually obtained by extracting the most mentioned entities from real-world KGs. We find that this policy leads to various forms of imbalances and *biases*. For instance, the datasets sampled from Freebase [17] tend to only feature people with nationality USA. Furthermore, the same biases observed in training are also present in validation and testing: this indirectly *incentivizes* models to incorporate the biases, as they can be instrumental to produce the correct predictions and boost the evaluation results. In short, our models are yielding the correct answers for the wrong reasons. For instance, in FB15k-237, which is considered the most reliable dataset [13], among the 1210 training facts and 151 test facts with relation *film_budget_currency*, 1164 and 146 facts respectively feature the same tail *USA_dollar*. We find that in those test facts, models always predict the correct tail on the first try when the answer is *USA_dollar*, whereas they never manage to guess if it is a different currency.

A few studies in the past have highlighted criticalities in LP benchmarks, mainly with regard to test leakage [18, 19] or of unnatural distributions and structures [20, 13]. Nonetheless, to the best of our knowledge the presence of straight-up biases had gone almost unnoticed so far. This motivates a systematic analysis of how they affect datasets and models, especially considering that KG embeddings have been shown to be just as vulnerable to biases as word embeddings [21, 22]. We report an accurate comparison with these other researches in Section 5.

We provide a formal definition of 3 types of data bias that can affect LP models. We focus on the 5 best-established LP datasets and estimate for each of them the number of test samples affected by each type of bias. We then conduct an extensive re-evaluation of 9 models representative for all the major families of LP systems, showing how removing the biased test facts affects their overall predictive performance. All the code and resources generated in our work are available at our public repository.[1]

The paper is organized as follows. In Section 2 we overview how embedding-based models perform the LP task; in Section 3 we discuss the main types of data bias we analyze in our work; in Section 4 we report our experimental findings on how they affect LP models; in Section 5 we present works related to ours and in Section 6 we provide concluding remarks.

[1]https://github.com/merialdo/research.lpbias

2. Link Prediction on Knowledge Graphs

We define any Knowledge Graph as $KG = (\mathcal{E}, \mathcal{R}, \mathcal{G})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{G} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of facts connecting the entities via relations. Each fact can be formulated as a triple $\langle h,\ r,\ t\rangle$, where h is the *head*, r is the *relation*, and t is the *tail*.

Most LP models nowadays map entities and relations to vectorized representations called *KG embeddings*. These models usually define a *scoring function* ϕ to estimate the plausibility of facts based on the embeddings of their elements. Embeddings are initialized randomly; then, they are trained with ML methods to optimize the scores of the known facts. When the training is over, the learned embeddings should be able to generalize and yield good ϕ values for unseen true facts as well. Models may also feature deep architectures of neural layers, which can be used in ϕ to process the embeddings of elements of the facts to score. The weights of neural layers are trained jointly with the KG embeddings.

Given a trained model, a *tail prediction* $\langle h,\ r,\ t\rangle$ is the process that finds t to be the best-scoring entity to complete the triple $\langle h,\ r,\ ?\rangle$: i.e., t is the answer to the question «What is the most likely tail for head h and relation r?»[2]:

$$t = \operatorname*{argmax}_{e \in \mathcal{E}} \phi(h, r, e). \tag{1}$$

A formulation for *head predictions* can be defined analogously.

LP research typically relies on datasets sampled from real-world KGs. Any dataset has its own sets $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{G}$, and $\mathcal{G}$ is usually split into a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{valid}$ and a test set $\mathcal{G}_{test}$. To evaluate the predictive performance of a model on a dataset, head and tail predictions are performed on each test fact in its $\mathcal{G}_{test}$. For each prediction, the *target entity* featured in the test fact (i.e., the expected prediction) is ranked against all other entities in $\mathcal{E}$. Given any test fact $\langle h,\ r,\ t\rangle$, the *tail rank* can be thus computed as:

$$tailRank(h, r, t) = |\{e \in \mathcal{E} | \phi(h, r, e) >= \phi(h, r, t)\}| \tag{2}$$

Head ranks can be computed analogously. An ideal model, or an *oracle*, would obtain rank 1 in the head and tail predictions of all test facts. The set T of all the head and tail ranks obtained in testing are then gathered into global metrics:

- *Hits@K (H@K)*: the fraction of ranks in T lesser or equal than a value k.
- *Mean Reciprocal Rank (MRR)*: the average of the inverse values of all the ranks in T.

Both $H@K$ and MRR are always between 0 and 1; the higher their value, the better the result they convey. These metrics can be computed in two separate settings: in the *raw setting* correct answers that outrank the target one are still deemed wrong; in the *filtered setting* they are not considered mistakes and do not contribute to rank computation. *Raw* metrics can sometimes be misleading, so *filtered* metrics are generally preferred in literature [16]; therefore, in this work we always use filtered metrics.

[2]In this formulation we assume higher ϕ scores convey better plausibility; analogous formulations are defined for models where higher scores convey worse plausibility.

3. Forms of Data Bias

In this section, we define 3 main types of bias commonly found in LP datasets. All of them are forms of *Sample Selection Bias* [23], i.e., unwanted, unrealistic patterns in a dataset caused by imbalances in the processes and sources used to construct it. We provide definitions from the perspective of a tail prediction $\langle h,\ r,\ t\rangle$; analogous definitions can be used for head predictions.

Type 1 Bias. A tail prediction $\langle h,\ r,\ t\rangle$ is prone to *Type 1 Bias* if the training facts mentioning r tend to always feature t as tail. For example, the tail prediction $\langle Barack\_Obama, gender, male\rangle$ is prone to this type of bias if the vast majority of gendered entities in the training set are males: this artificially favours the prediction of male genders. In practice, we verify if the fraction between the number of training facts featuring both r and t and the number of training facts featuring r exceeds a threshold τ_1. In our experiments we set $\tau_1 = 0.75$.

Type 2 Bias. A tail prediction $\langle h,\ r,\ t\rangle$ in which r is a one-to-many or a many-to-many relation is prone to *Type 2 Bias* if, whenever an entity e is seen as head for relation r, fact $\langle e,\ r,\ t\rangle$ also exists in $\mathcal{G}_{train}$. Type 2 Bias affects relations that have a "default" correct answer. Differently from Type 1, facts mentioning r may feature a variety of tails different from t; however, for each entity e seen as head these facts, t tends to always be among the correct tails too. This makes $\langle e,\ r,\ t\rangle$ artificially easier to predict. For instance, the tail prediction $\langle Cristiano\_Ronaldo, language, English\rangle$ is prone to Type 2 Bias if most people, in addition to other languages, also speak *English*. In practice, we verify if the fraction of entities e seen as heads for relation r and that also display a fact $\langle e,\ r,\ t\rangle$ exceeds a threshold τ_2. In our experiments we use $\tau_2 = 0.5$.

Type 3 Bias. A tail prediction $\langle h,\ r,\ t\rangle$ is prone to *Type 3 Bias* if a relation s exists such that: *(i)* whenever s links two entities, r links them as well; and *(ii)* the fact $\langle h,\ s,\ t\rangle$ is present in the training set. For example, in the FB15k dataset the producer of a TV program is almost always its creator too; this may lead to assume that creating a program implies being its producer. In practice, to verify if s and r share this correlation we check if the fraction of s mentions in which s also co-occurs with r is greater than a threshold τ_3. In our experiments we set $\tau_3 = 0.5$.

4. Data Bias in Link Prediction Benchmarks

We discuss in this section our experimental findings on the three forms of data bias defined in Section 3. We first describe how these biases affect LP datasets; we then analyze their consequences on the behaviour of LP models.

4.1. Data Biases in LP Datasets

We take into account the 5 most popular LP datasets in literature: FB15k, WN18, FB15k-237, WN18RR and YAGO3-10. **FB15k** and **WN18** have been built by Bordes *et al.* [16] by selecting the facts featuring the richest entities in Freebase [17] and WordNet [24]. Toutanova and Chen [18] and Dettmers *et al.* [19] have observed that such datasets suffer from test leakage due to the pervasive presence of inverse and equivalent relations; they have filtered away such relations to create the more challenging subsamples **FB15k-237** and **WN18RR**. Finally, Dettmers *et al.* [19]

	Entities	Relations	Facts			Test Predictions				
			Train	Valid	Test	All	w/o B1	w/o B2	w/o B3	w/o B∗
FB15k	14.951	1.345	483.142	50.000	59.071	118.142	115.165 (-2.5%)	108.705 (-8.0%)	101.406 (-14.2%)	93.005 (-21.3)
FB15k-237	14.541	237	272.115	17.535	20.466	40.932	39.145 (-4.4%)	36.482 (-5.5%)	40.932 (-0.0%)	36.912 (-9.8%)
WN18	40.943	18	141.442	5.000	5.000	10.000	10.000 (-0.0%)	10.000 (-0.0%)	10.000 (-0.0%)	10.000 (-0.0%)
WN18RR	40.943	11	86.835	3.034	3.134	6.268	6.268 (-0.0%)	6.268 (-0.0%)	6.268 (-0.0%)	6.268 (-0.0%)
YAGO3-10	123.155	37	1.078.868	5.000	5.000	10.000	9.725 (-2.8%)	9.992 (-0.1%)	4.876 (-51.2%)	4.593 (-54.1%)

Table 1
Dataset statistics and numbers of test predictions prone to bias.

have also created the **YAGO3-10** dataset selecting the facts featuring entities linked by at least 10 different relations in the YAGO3 KG [25].

In these datasets the training, validation and test sets are sampled from the same distributions, so any bias seen in training will also be featured in validation and testing. For each test prediction in each dataset we verify if it is prone to any type of bias applying the definitions in Section 3. Table 1 reports the main statistics and the number of test predictions not affected by Type 1 Bias (w/o B1), Type 2 Bias (w/o B2), Type 3 Bias (w/o B3), and by any type of bias at all (w/o B⋆). Note that these numbers are not cumulative because the same prediction may be affected by multiple types of bias.

The most affected dataset is YAGO3-10, with 54.1% of its test facts being prone to bias, especially of Type 3. This confirms the finding of Akrami *et al* [13] that the two most common relations in the dataset (*affiliated_to* and *plays_for*) are almost interchangeable. FB15k and FB15k-237 are noticeably affected too, with 26.7% and 9.8% of their test facts prone to bias respectively. The Type 3 Bias is not present at all in FB15k-237; this is not surprising, because this dataset was built by design with no inverse or equivalent relations. WN18 and WN18RR, finally, seem completely immune to any type of bias; this is possibly due the nature of WordNet, which is a lexical ontology rather than a KG.

4.2. Data Biases and LP models

In this section we study how data bias affects the behaviour of LP models. As mentioned in Section 4.1, LP benchmarks display the same biases across their training and test sets. Hence, in addition to being exposed to bias, unbeknownst to developers, LP models are actually encouraged to incorporate it, because such bias can help to identify the correct answers in testing. In the long run this may favour the architectures most vulnerable to bias over the most robust ones.

To assess the severity of this condition we remove from each dataset the test predictions prone to each type of bias, and measure how this affects the evaluation results of LP models relying on diverse architectures. We focus on H@1 and MRR, usually considered the most characterizing metric in LP. We stress that we do not modify the training sets of the datasets: we just filter away the bias-prone test predictions, in the quantities reported in Table 1.

We use as a starting point the evaluation results of 19 LP models publicly shared by Rossi *et al.* [14]. Due to space constraints, we report our findings on 9 models represent-

			FB15k					FB15k-237					YAGO3-10				
			Orig.	w/o B1	w/o B2	w/o B3	w/o B*	Orig.	w/o B1	w/o B2	w/o B3	w/o B*	Orig.	w/o B1	w/o B2	w/o B3	w/o B*
Matrix Decomposition	ComplEx	H@1	0.823	0.819 (-0.6%)	0.814 (-1.1%)	0.799 (-2.9%)	0.788 (-4.3%)	0.272	0.239 (-12.0%)	0.241 (-11.3%)	0.272 (0.0%)	0.205 (-24.6%)	0.501	0.487 (-2.7%)	0.501 (-0.1%)	0.232 (-53.7%)	0.186 (-62.9%)
		MRR	0.855	0.851 (-0.4%)	0.846 (-1.0%)	0.835 (-2.4%)	0.824 (-3.6%)	0.367	0.338 (-7.8%)	0.337 (-8.1%)	0.367 (0.0%)	0.306 (-16.7%)	0.577	0.566 (-2.0%)	0.577 (-0.1%)	0.307 (-46.8%)	0.265 (-54.1%)
	TuckER	H@1	0.729	0.722 (-0.9%)	0.716 (-1.8%)	0.705 (-3.3%)	0.689 (-5.5%)	0.259	0.225 (-13.0%)	0.228 (-12.0%)	0.259 (0.0%)	0.191 (-26.3%)	0.466	0.460 (-1.1%)	0.466 (0.0%)	0.186 (-60.1%)	0.158 (-66.1%)
		MRR	0.788	0.783 (-0.7%)	0.777 (-1.4%)	0.767 (-2.7%)	0.754 (-4.4%)	0.352	0.323 (-8.4%)	0.322 (-8.5%)	0.352 (0.0%)	0.290 (-17.8%)	0.544	0.537 (-1.3%)	0.544 (0.0%)	0.257 (-52.7%)	0.225 (-58.6%)
Geometric	TransE	H@1	0.494	0.492 (-0.3%)	0.467 (-5.3%)	0.463 (-6.2%)	0.438 (-11.2%)	0.217	0.184 (-15.3%)	0.189 (-12.8%)	0.217 (0.0%)	0.153 (-29.6%)	0.406	0.390 (-4.0%)	0.406 (0.0%)	0.135 (-66.8%)	0.083 (-79.4%)
		MRR	0.628	0.624 (-0.6%)	0.608 (-3.1%)	0.597 (-4.9%)	0.577 (-8.1%)	0.310	0.280 (-9.7%)	0.282 (-9.1%)	0.310 (0.0%)	0.249 (-19.8%)	0.501	0.487 (-2.8%)	0.501 (0.0%)	0.219 (-56.2%)	0.172 (-65.6%)
	CrossE	H@1	0.601	0.592 (-1.5%)	0.590 (-1.8%)	0.573 (-4.6%)	0.561 (-6.6%)	0.212	0.178 (-15.9%)	0.182 (-14.2%)	0.212 (0.0%)	0.145 (-31.7%)	0.331	0.321 (-3.0%)	0.33 (-0.1%)	0.128 (-61.4%)	0.093 (-71.8%)
		MRR	0.702	0.695 (-1.0%)	0.693 (-1.3%)	0.677 (-3.6%)	0.666 (-5.1%)	0.298	0.267 (-10.3%)	0.267 (-10.2%)	0.298 (0.0%)	0.233 (-21.6%)	0.446	0.435 (-2.5%)	0.445 (-0.1%)	0.207 (-53.7%)	0.167 (-62.4%)
	HAKE	H@1	0.745	0.739 (-0.8%)	0.734 (-1.5%)	0.721 (-3.2%)	0.708 (-4.9%)	0.249	0.218 (-12.5%)	0.223 (-10.6%)	0.249 (0.0%)	0.189 (-24.2%)	0.463	0.453 (-2.2%)	0.462 (0.0%)	0.199 (-57.1%)	0.161 (-65.2%)
		MRR	0.796	0.791 (-0.6%)	0.786 (-1.2%)	0.775 (-2.6%)	0.763 (-4.1%)	0.347	0.319 (-8.1%)	0.320 (-7.6%)	0.347 (0.0%)	0.289 (-16.5%)	0.546	0.536 (-1.9%)	0.546 (0.0%)	0.273 (-50.0%)	0.234 (-57.2%)
Deep Learning	InteractE	H@1	0.726	0.719 (-1.0%)	0.711 (-2.1%)	0.695 (-4.2%)	0.677 (-6.7%)	0.263	0.231 (-12.5%)	0.233 (-11.6%)	0.263 (0.0%)	0.197 (-25.3%)	0.466	0.451 (-3.2%)	0.466 (-0.1%)	0.213 (-54.4%)	0.165 (-64.7%)
		MRR	0.786	0.780 (-0.7%)	0.774 (-1.5%)	0.760 (-3.3%)	0.745 (-5.2%)	0.355	0.326 (-8.2%)	0.325 (-8.4%)	0.355 (0.0%)	0.293 (-17.4%)	0.543	0.531 (-2.4%)	0.543 (-0.1%)	0.277 (-49.1%)	0.232 (-57.2%)
	CapsE	H@1	0.019	0.015 (-23.7%)	0.014 (-29.4%)	0.011 (-40.9%)	0.007 (-66.0%)	0.073	0.040 (-45.5%)	0.064 (-13.4%)	0.073 (0.0%)	0.028 (-62.1%)	0.000	0.000 (0.0%)	0.000 (0.0%)	0.000 (0.0%)	0.000 (0.0%)
		MRR	0.087	0.075 (-13.0%)	0.073 (-15.5%)	0.072 (-16.5%)	0.056 (-35.9%)	0.160	0.126 (-21.1%)	0.145 (-9.6%)	0.160 (0.0%)	0.108 (-32.4%)	0.000	0.000 (0.0%)	0.000 (0.0%)	0.000 (0.0%)	0.000 (0.0%)
	RSN	H@1	0.723	0.717 (-0.9%)	0.71 (-1.9%)	0.690 (-4.6%)	0.674 (-6.8%)	0.198	0.163 (-17.8%)	0.169 (-15.0%)	0.198 (0.0%)	0.130 (-34.6%)	0.492	0.477 (-2.9%)	0.426 (-0.1%)	0.164 (-61.6%)	0.118 (-72.3%)
		MRR	0.777	0.771 (-0.7%)	0.765 (-1.6%)	0.748 (-3.7%)	0.734 (-5.6%)	0.280	0.247 (-11.5%)	0.249 (-10.9%)	0.280 (0.0%)	0.214 (-23.6%)	0.560	0.547 (-2.2%)	0.51 (-0.1%)	0.231 (-54.8%)	0.187 (-63.5%)
	AnyBURL	H@1	0.815	0.810 (-0.5%)	0.804 (-1.3%)	0.795 (-2.4%)	0.782 (-4.0%)	0.269	0.236 (-12.2%)	0.237 (-12.0%)	0.269 (0.0%)	0.201 (-25.5%)	0.492	0.477 (-2.9%)	0.491 (-0.1%)	0.218 (-55.7%)	0.169 (-65.6%)
		MRR	0.837	0.833 (-0.4%)	0.827 (-1.2%)	0.818 (-2.2%)	0.806 (-3.7%)	0.353	0.324 (-8.2%)	0.322 (-8.9%)	0.353 (0.0%)	0.290 (-18.0%)	0.560	0.547 (-2.2%)	0.559 (-0.1%)	0.282 (-49.5%)	0.238 (-57.4%)

Table 2
Evaluation results on FB15k, FB15k-237 and YAGO3-10 before and after removing the test predictions prone to bias.

ing all the families in their work: ComplEx [26] and TuckER [27] for the Matrix Decomposition models; TransE [16], CrossE [28] and HAKE [29] for the Geometric models; InteractE [30], RSN [31] and CapsE [32] for the Deep Learning models; and the rule-based AnyBURL [33] as a baseline. Complete results for all the 19 models are available in our online repository.

Table 2 shows the results for datasets FB15k, FB15k-237 and YAGO3-10. If models were immune to biases, removing bias-prone test predictions should not heavily affect their metrics. On the contrary, we observe impressive performance drops across all models. YAGO3-10 results display the largest worsening, with models losing 50% to 70% of their H@1 and MRR. In FB15k-

			WN18					WN18RR				
			Orig.	w/o B*	w/o inverse	w/o symmetric	w/o inverse or symmetric	Orig.	w/o B*	w/o inverse	w/o symmetric	w/o inverse or symmetric
Matrix Decomposition	ComplEx	H@1	0.944	0.944 (0.0%)	0.807 (-14.5%)	0.929 (-1.6%)	0.215 (-77.2%)	0.443	0.443 (0.0%)	0.443 (0.0%)	0.161 (-63.6%)	0.161 (-63.6%)
		MRR	0.951	0.951 (0.0%)	0.827 (-13.0%)	0.937 (-1.4%)	0.295 (-69.0%)	0.489	0.489 (0.0%)	0.489 (0.0%)	0.230 (-52.9%)	0.230 (-52.9%)
	TuckER	H@1	0.946	0.947 (0.0%)	0.812 (-14.2%)	0.932 (-1.5%)	0.232 (-75.4%)	0.429	0.429 (0.0%)	0.429 (0.0%)	0.141 (-67.1%)	0.141 (-67.1%)
		MRR	0.951	0.951 (0.0%)	0.826 (-13.1%)	0.938 (-1.4%)	0.290 (-69.5%)	0.459	0.459 (0.0%)	0.459 (0.0%)	0.185 (-59.6%)	0.185 (-59.6%)
Geometric	TransE	H@1	0.406	0.406 (0.0%)	0.009 (-97.8%)	0.514 (+26.6%)	0.037 (-91.0%)	0.028	0.028 (0.0%)	0.028 (0.0%)	0.042 (+50.5%)	0.042 (+50.5%)
		MRR	0.646	0.646 (0.0%)	0.334 (-48.4%)	0.713 (+10.2%)	0.135 (-79.1%)	0.206	0.206 (0.0%)	0.206 (0.0%)	0.110 (-46.5%)	0.110 (-46.5%)
	CrossE	H@1	0.733	0.732 (0.0%)	0.774 (+5.6%)	0.668 (-8.9%)	0.148 (-79.8%)	0.381	0.381 (0.0%)	0.381 (0.0%)	0.073 (-80.8%)	0.073 (-80.8%)
		MRR	0.834	0.834 (0.0%)	0.797 (-4.5%)	0.793 (-4.9%)	0.208 (-75.1%)	0.405	0.405 (0.0%)	0.405 (0.0%)	0.106 (-73.7%)	0.106 (-73.7%)
	HAKE	H@1	0.943	0.944 (0.0%)	0.803 (-14.9%)	0.928 (-1.6%)	0.196 (-79.2%)	0.453	0.453 (0.0%)	0.453 (0.0%)	0.177 (-61.0%)	0.177 (-61.0%)
		MRR	0.950	0.950 (0.0%)	0.823 (-13.4%)	0.936 (-1.4%)	0.278 (-70.7%)	0.497	0.497 (0.0%)	0.497 (0.0%)	0.243 (-51.1%)	0.243 (-51.1%)
Deep Learning	InteractE	H@1	0.946	0.946 (0.0%)	0.811 (-14.3%)	0.932 (-1.5%)	0.232 (-75.4%)	0.427	0.427 (0.0%)	0.427 (0.0%)	0.138 (-67.7%)	0.138 (-67.7%)
		MRR	0.950	0.949 (0.0%)	0.821 (-13.5%)	0.936 (-1.4%)	0.273 (-71.3%)	0.458	0.458 (0.0%)	0.458 (0.0%)	0.184 (-59.8%)	0.184 (-59.8%)
	CapsE	H@1	0.846	0.846 (0.0%)	0.699 (-17.4%)	0.828 (-2.1%)	0.042 (-95.0%)	0.337	0.337 (0.0%)	0.337 (0.0%)	0.071 (-79.0%)	0.071 (-79.0%)
		MRR	0.890	0.890 (0.0%)	0.748 (-15.9%)	0.873 (-1.8%)	0.123 (-86.2%)	0.415	0.415 (0.0%)	0.415 (0.0%)	0.158 (-61.9%)	0.158 (-61.9%)
	RSN	H@1	0.912	0.912 (0.0%)	0.773 (-15.2%)	0.895 (-1.9%)	0.142 (-84.5%)	0.346	0.346 (0.0%)	0.346 (0.0%)	0.074 (-78.5%)	0.074 (-78.5%)
		MRR	0.928	0.928 (0.0%)	0.793 (-14.6%)	0.912 (-1.7%)	0.190 (-79.5%)	0.395	0.395 (0.0%)	0.395 (0.0%)	0.124 (-68.7%)	0.124 (-68.7%)
	AnyBURL	H@1	0.947	0.947 (0.0%)	0.815 (-13.9%)	0.933 (-1.5%)	0.247 (-73.9%)	0.458	0.458 (0.0%)	0.458 (0.0%)	0.184 (-59.9%)	0.184 (-59.9%)
		MRR	0.953	0.953 (0.0%)	0.833 (-12.6%)	0.94 (-1.3%)	0.318 (-66.6%)	0.497	0.497 (0.0%)	0.497 (0.0%)	0.243 (-51.1%)	0.243 (-51.1%)

Table 3
Variations in the evaluation results on WN18 and WN18RR after removing the test predictions prone to bias or influenced by inverse and symmetric relations.

237, often considered the most reliable dataset, their decrease is still between 20% and 35%. In FB15k they lose around 5% of the original metrics, but this apparent robustness is likely due to the presence of inverse relations facilitating predictions.

Table 3 reports the results for WN18 and WN18RR. As already described in Section 4.1, in these datasets test predictions do not appear prone to bias. Nonetheless this does not make them more reliable; on the contrary, their test predictions seem only enabled by the presence of symmetric and (in WN18) inverse relations. Table 3 also displays that removing the test predictions featuring symmetric or inverse relations results in plummeting H@1 and MRR values, with a decrease usually between 50% and 75%.

4.3. Key Takeaways

We find that the policies used to generate LP datasets have led to severe forms of selection bias; this, in turn, has made significant fractions of their test sets artificially easier to predict than the others. Our results prove that LP models are indeed sensitive to these forms of bias, as filtering away the affected test predictions heavily worsens their evaluation metrics. Both neural and rule-based LP models appear equally vulnerable to this phenomenon: this proves that the issue is not rooted in the technique used to learn the facts, but rather in the data sources themselves. Not all datasets are interested by this condition to the same extent. YAGO3-10 and FB15k-237 are the most affected, and show the heaviest drop in performance when removing the biased test facts. Wordnet-based datasets, on the other hand, do not display bias at all.

The bias problem, in itself, can probably be avoided by just skipping the test facts prone to bias during evaluation. However, suggesting to just adopt this practice would be naive from our part. Quite worryingly, even in bias-free datasets we observe that most correct predictions are just enabled by the presence of inverse and symmetric relations. In other words, in *all* datasets, when removing the test predictions affected by either bias or inverse/symmetric relations, the predictive performance of models plummets.

This makes us wonder how many of the remaining test facts are actually predictable. If we just removed the bias-prone test predictions, we may mostly end up with test sets that not even humans, with the information available in training, can infer; if this was the case, the whole task would become pointless. We intend to conduct further studies in this regard, asking this question directly to human workers. If the outcome will prove that most non-biased test facts are indeed unpredictable, then it will be painfully necessary to replace the current datasets with novel ones extracted in more sensible ways, possibly keeping humans in the loop for the selection of training and test facts.

5. Related Works

The works most related to ours consist in analyses that highlight criticalities of the current LP benchmarking techniques. So far, compared to the wide body of literature proposing new LP models, a relatively small effort has been devoted to analyzing their evaluation practices.

Toutanova and Chen [18] have been the first to notice the presence of test leakage in FB15k due to inverse relations. They have assessed the severity of the issue by proving that a simple model based on observable features achieves competitive performance on the dataset; they have proceeded to remove these relations from FB15k generating the more challenging FB15k-237.

A similar study has been carried out by Dettmers *et al.* [19] on FB15k and WN18, showing that a trivial system based on inverse relations can achieve state-of-the-art results on both datasets; the authors have then generated WN18RR as a more challenging subsample of WN18.

Akrami *et al.* [13] have carried out an extensive analysis quantifying the effect of various artificial patterns in LP datasets, such as inverse relations and Cartesian product relations. In addition to confirming the above mentioned observations on FB15k and WN18 they have found that LP performances are boosted in FB15k and FB15k-237 by the redundant structures of Cartesian product relations, and in YAGO3-10 by the presence of equivalent relations.

Nayyeri *et al.* [34] refer to KGs in which facts also feature additional numerical weights to convey various meanings, e.g., the confidence of each fact. They acknowledge that the presence of bias in these values hinders the effectiveness of the learned embeddings, and propose a Weighted Triple Loss that, while taking advantage of these weights, is also robust to their biases.

Rossi and Matinata [20] have shown that the distributions of entities in all LP datasets are wildly skewed: a few entities are featured in thousands of training facts, making them easier to learn and predict, whereas the others may only occur a handful of times. The rich and easily predictable entities are over-represented in testing, thus affecting fairness of such benchmarks.

The same concerns have been shared by Mohamed *et al.* [35]; to overcome this issue, they have proposed novel stratified versions of the Hits@K and MRR metrics, called Strat-Hits@K and Strat-MRR respectively. These metrics should estimate of the predictive performance of models in a fairer way, unbiased by the popularity over-represented entities.

All these works share the same spirit of ours in their goal to identify the shortcomings of current LP evaluation approaches, in order to drive research towards more realistic and healthy practices. Our main difference lies in our definition and identification of *bias structures* that had so far gone unnoticed, as well as our systematic methodology of re-computing the predictive performances of a wide variety of models after removing the biased test facts.

6. Conclusion

We have reported an analysis on the presence of bias across the 5 best-established datasets for Link Prediction on KGs. We have defined 3 main types of Selection Sample Bias, and we have observed that they affect significant portions of the test predictions in 3 datasets out of 5. We have then analyzed how removing such bias-prone predictions alters the evaluation results of 9 models representing the main families of Link Prediction systems. The result is generally a significant drop in predictive performance. This proves that a large part of the correct predictions output by models on those datasets is indeed facilitated by the presence of bias. The very low values obtained on this de-biased test scenario suggests that many of the remaining test facts may not be predictable at all.

We thus call for the production of more effective and robust datasets for Link Prediction, and for the definition of more thorough evaluation methods that take into account their properties.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The semantic web, Springer, 2007.

[2] T. P. Tanon, G. Weikum, F. M. Suchanek, YAGO 4: A reason-able knowledge base, in: ESWC, 2020.

[3] D. Vrandecic, M. Krötzsch, Wikidata: a free collaborative knowledge base, CACM (2014).

[4] E. Hovy, R. Navigli, S. P. Ponzetto, Collaboratively Built Semi-structured Content and Artificial Intelligence: The Story So Far, Artif. Intell. (2013).

[5] W. Yih, M. Chang, X. He, J. Gao, Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base, in: ACL, 2015.

[6] A. Singhal, Introducing the knowledge graph: things, not strings, 2012. URL: https://www.blog.google/products/search/introducing-knowledge-graph-things-not/, blogpost in the Official Google Blog.

[7] R. Qian, Understand your world with bing, 2013. URL: https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/, blogpost in Bing Blogs.

[8] X. L. Dong, Building a broad knowledge graph for products, in: ICDE, 2019.

[9] R. Pittman, Cracking the code on conversational commerce, 2017. URL: https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/, blogpost in Ebay Inc. Stories.

[10] T. Stocky, L. Rasmussen, Introducing graph search beta, 2014. URL: https://newsroom.fb.com/news/2013/01/introducing-graph-search-beta/, blogpost in Facebook Newsroom.

[11] Q. He, B.-C. Chen, D. Agarwal, Building the linkedin knowledge graph, 2016. URL: https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph/, blogpost in LinkedIn Blog.

[12] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, D. Lin, Knowledge base completion via search-based question answering, in: WWW, 2014.

[13] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, C. Li, Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study, in: SIGMOD, 2020.

[14] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: A comparative analysis, ACM TKDD (2021).

[15] M. Wang, L. Qiu, X. Wang, A survey on knowledge graph embeddings for link prediction, Symmetry (2021).

[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NIPS, 2013.

[17] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: SIGMOD, 2008.

[18] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, in: CVSC, 2015.

[19] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: AAAI, 2018.

[20] A. Rossi, A. Matinata, Knowledge Graph Embeddings: Are Relation-Learning Models Learning Relations?, in: PIE, 2020.

[21] J. Fisher, D. Palfrey, C. Christodoulopoulos, A. Mittal, Measuring social bias in knowledge graph embeddings, arXiv preprint arXiv:1912.02761 (2019).

[22] S. Bourli, E. Pitoura, Bias in knowledge graph embeddings, in: ASONAM, IEEE, 2020.

[23] J. J. Heckman, Sample selection bias as a specification error, Econometrica (1979).

[24] G. A. Miller, Wordnet: a lexical database for english, CACM (1995).

[25] F. Mahdisoltani, J. Biega, F. M. Suchanek, YAGO3: A knowledge base from multilingual wikipedias, in: CIDR, 2015.

[26] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex Embeddings for Simple Link Prediction, in: ICML, 2016.

[27] I. Balazevic, C. Allen, T. M. Hospedales, TuckER: Tensor Factorization for Knowledge Graph Completion, in: EMNLP - IJCNLP, 2019.

[28] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, H. Chen, Interaction embeddings for prediction and explanation in knowledge graphs, in: WSDM, 2019.

[29] Z. Zhang, J. Cai, Y. Zhang, J. Wang, Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction, in: AAAI, 2020.

[30] S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, P. P. Talukdar, InteractE: Improving Convolution-based Knowledge Graph Embeddings by Increasing Feature Interactions, in: AAAI, 2020.

[31] L. Guo, Z. Sun, W. Hu, Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs, in: ICML, 2019.

[32] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, D. Q. Phung, A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization, in: NAACL-HLT, 2019.

[33] C. Meilicke, M. W. Chekol, M. Fink, H. Stuckenschmidt, Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion, arXiv preprint arXiv:2004.04412 (2020).

[34] M. Nayyeri, G. M. Cil, S. Vahdati, F. Osborne, A. Kravchenko, S. Angioni, A. A. Salatino, D. R. Recupero, E. Motta, J. Lehmann, Link prediction of weighted triples for knowledge graph completion within the scholarly domain, IEEE Access (2021).

[35] A. Mohamed, S. Parambath, Z. Kaoudi, A. Aboulnaga, Popularity agnostic evaluation of knowledge graph embeddings, in: UAI, PMLR, 2021.

Integrating Contextual Knowledge to Visual Features for Fine Art Classification

Giovanna Castellano, Giovanni Sansaro and Gennaro Vessio

Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Abstract

Automatic art analysis has seen an ever-increasing interest from the pattern recognition and computer vision community. However, most of the current work is mainly based solely on digitized artwork images, sometimes supplemented with some metadata and textual comments. A knowledge graph that integrates a rich body of information about artworks, artists, painting schools, etc., in a unified structured framework can provide a valuable resource for more powerful information retrieval and knowledge discovery tools in the artistic domain. To this end, this paper presents *ArtGraph*: an artistic knowledge graph based on WikiArt and DBpedia. The graph, implemented in Neo4j, already provides knowledge discovery capabilities without having to train a learning system. In addition, the embeddings extracted from the graph are used to inject "contextual" knowledge into a deep learning model to improve the accuracy of artwork attribute prediction tasks.

Keywords

digital humanities, visual arts, knowledge graphs, deep learning

1. Introduction

In recent years, Knowledge Graphs (KGs) have emerged as a powerful tool for describing real-world entities and their relationships, and are increasingly used for many practical tasks, from recommendations to risk assessment [1]. At the same time, the last decade has seen a remarkable range of advances in Machine Learning—and particularly in Deep Learning (DL) approaches based on neural networks [2]—, to build ever more accurate systems in a wide range of areas, particularly computer vision and natural language processing. Combining the expressiveness of KGs with the learning ability of deep neural networks promises to develop even more effective algorithms for many *downstream* tasks.

One of the many domains that can benefit from using KGs in conjunction with DL solutions is the artistic one. Leveraging DL algorithms in this domain, particularly Convolutional Neural Network (CNN) models, has already proven effective in tackling several challenging tasks, from object detection in paintings to style classification [3]. And this success is mainly due to the growing availability of large digitized fine art collections, such as WikiArt.[1] However, while promising, most of the existing solutions rely solely on the visual features that a CNN can automatically extract from digital images of paintings, drawings, etc. (e.g., [4, 5, 6]). This

DL4KG'21: Workshop on Deep Learning for Knowledge Graphs

giovanna.castellano@uniba.it (G. Castellano); gennaro.vessio@uniba.it (G. Vessio)

0000-0002-6489-8628 (G. Castellano); 0000-0002-1282-6657 (G. Sansaro); 0000-0002-0883-2691 (G. Vessio)

[1]https://www.wikiart.org/

inevitably leads to the neglect of an enormous amount of knowledge—already available from disparate sources—, relating to the "context" of each artwork. An artwork, in fact, is characterized not only by its visual appearance, but also by various other historical, social and contextual factors that place it in a much more complex and multifaceted scenario.

A promising way to harness this knowledge to improve the accuracy of art-based analytic systems is to encode the contextual information of the artworks into a KG and use an appropriate representation of the nodes in the graph, for example by means of *embeddings* [7], as a novel, additional input to a deep learning model. Our research goes in this direction.

Related Work For the sake of brevity, we limit our review of the related literature only to the work most directly linked to the research presented here. The interested reader can refer to our recent review article [3] for a broader view on the computational analysis of art. This paper is inspired by research conducted by Garcia et al. [8]. They combined a multi-output model trained to solve attribute prediction tasks based on visual features with a second model based on non-visual information extracted from artistic metadata encoded using a KG. This model was intended to inject "context" information to improve the performance of the first model. The general framework was called *ContextNet*. To encode the KG information into a vector representation, the popular node2vec model [9] was adopted. The KG was built using only the information provided by SemArt, a dataset previously proposed in [10] that provides not only artwork images and their attributes, but also artistic comments intended to achieve semantic art understanding. However, metadata are only available for artworks in the dataset, so adding a new artwork would not result in any domain information about it. In addition, the proposed graph has the artist node, which allows to connect artworks with the same artist, but without considering the relationships between artists, such as artistic influence.

Our Contribution The two limitations mentioned above can be overcome by relying on a source of knowledge external to the dataset, such as Wikipedia, which provides an enormous amount of information, even in a structured form. Furthermore, the KG could not be treated only as an adjacency matrix from which embeddings can be extracted as auxiliary information to be provided to learning models. Instead, the KG can be encoded into a NoSQL database, such as Neo4j, which can already help provide a powerful knowledge discovery framework without explicitly training a learning system. In this paper we present *ArtGraph*, an artistic knowledge graph. The proposed KG integrates information collected by WikiArt and DBpedia, and exploits the potential of the Neo4j database management system, which provides an expressive modeling and graph query language. The proposed KG encodes a broad representation of the artistic domain, with multiple metadata and relationships between artists. Also, we propose a novel approach to inject contextual knowledge into a deep network.

2. *ArtGraph*

ArtGraph is a KG in the art domain capable of representing and describing concepts related to artworks. Our KG can represent a wide range of relationships, including those between artists and their works. A comparison between our proposed KG and the one presented by Garcia et

Table 1
Comparison between our KG and the one proposed by Garcia et al. [8]. It is worth noting that, although SemArt has more than 3000 unique artists, most of them are associated with fewer than ten artworks.

KG	# nodes	# edges	# artists	# artworks	# relations btw artworks	# relations btw artists
ContextNet	33,148	125,506	3166	19,244	7	0
ArtGraph	74,382	537,883	300	63,145	10	7

al. is provided in Table 1. It is worth noting that, at the current stage of our research, we are only focusing on (the most popular) 300 artists, as we are interested in a richer representation of the relationships between them and other entities.

The metadata extracted from WikiArt have been transformed into relationships and nodes mainly related to the artworks, their genre, style, location, etc. Furthermore, since WikiArt does not provide rich information about artists, each artist of our KG is linked not only to the artworks produced but also to other nodes built using RDF triples extracted from DBpedia. Extracting and integrating data from these two sources required a laborious process of data cleaning and normalization, as well as manual intervention to resolve several inconsistencies among the data. Overall, the conceptual scheme of *ArtGraph* (represented in Fig. 1) includes *artwork* nodes and *artist* nodes:

- Each *artwork* node is connected to the following nodes: *tags* (e.g., woman, sea, birds), *genre* (e.g., self-portrait), *style*, *period*, *series* (e.g., "The Seasons" by Giuseppe Arcimboldo), *auction*, *media* (e.g., paper, watercolor), the *gallery* in which the artwork is located, and the *city* (or *country*) in which the artwork has been completed.
- Each *artist* node is connected to the following nodes: *field* (e.g., drawing, sculpture), *movement* (e.g., Surrealism, Renaissance, Pop Art), *training* (e.g., Accademia di Belle Arti di Firenze), *Wikipedia categories* (e.g., living people, people from Florence), other artists (*influences* or *teaching*, and *patrons*).

This structure allows the creation of a network between artists, which is useful for further analysis. In total, the resulting KG contains 74,382 nodes and 537,883 edges, with 300 artists, 63,145 artworks, 81 genres, 49 styles, and a huge plethora of metadata and textual comments describing them (Table 1).

ArtGraph has been implemented in Neo4j[2] on an i5-10400 system, with a 2.90 GHz CPU and 16GB of RAM. We preferred Neo4j to other existing solutions as it is a native graph database that provides a powerful and flexible framework for storing and querying graph-like structures. Using Neo4j, connections between data are stored and not calculated at query time. Cypher, which is the declarative query language adopted by Neo4j, takes advantage of these stored connections to provide an expressive and optimized language for graphs to execute even complex queries extremely quickly.

To allow for a visual exploration of the graph, we have created a web interface that uses JavaScript to connect to Neo4j (Fig. 2). The goal is to provide the end user—as mentioned above,

[2]https://neo4j.com

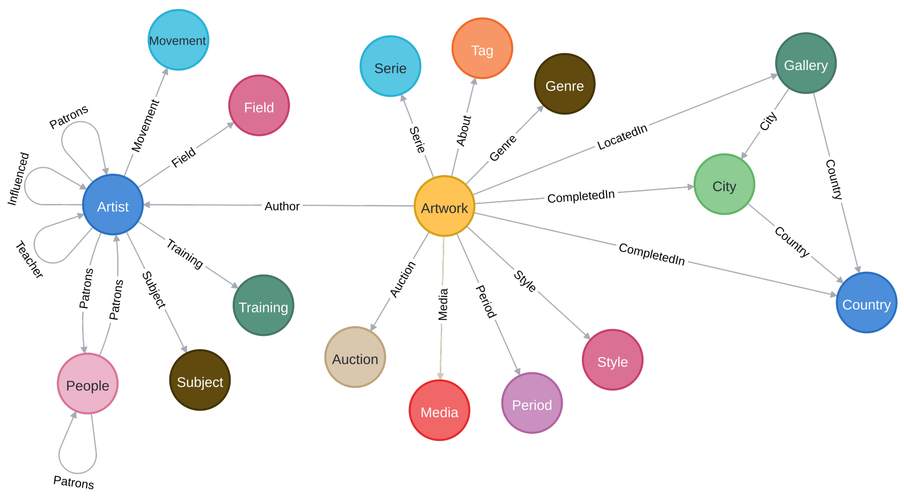
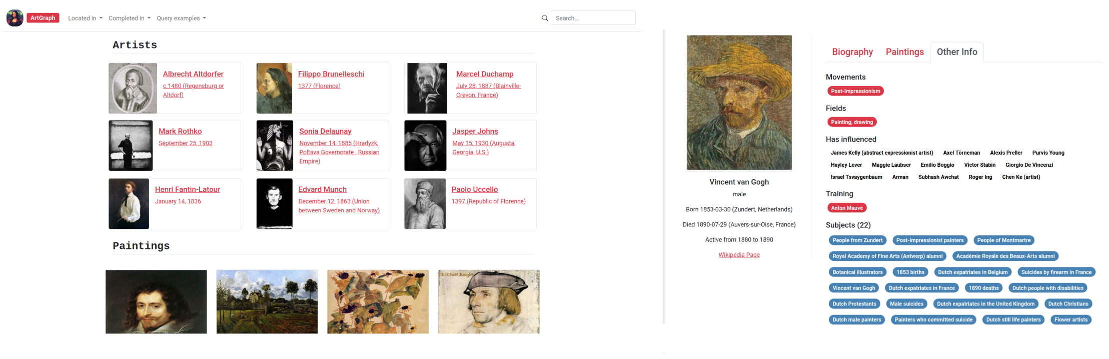


Figure 1: Scheme of *ArtGraph*. The nodes correspond to relevant entities in the artistic domain, while the edges represent existing relationships between them.



Artists

Albrecht Altdorfer
c.1480 (Regensburg or Altdorf)

Filippo Brunelleschi
1377 (Florence)

Marcel Duchamp
July 28, 1887 (Blainville-Crevon, France)

Mark Rothko
September 25, 1903

Sonia Delaunay
November 14, 1885 (Hradyzk, Poltava Governorate , Russian Empire)

Jasper Johns
May 15, 1930 (Augusta, Georgia, U.S.)

Henri Fantin-Latour
January 14, 1836

Edvard Munch
December 12, 1863 (Union between Sweden and Norway)

Paolo Uccello
1397 (Republic of Florence)

Paintings

Vincent van Gogh
male
Born 1853-03-30 (Zundert, Netherlands)
Died 1890-07-29 (Auvers-sur-Oise, France)
Active from 1880 to 1890
Wikipedia Page

Movements
Post-Impressionism

Fields
Painting, drawing

Has influenced
James Kelly (abstract expressionist artist) Axel Törneman Alexis Preller Purvis Young
Hayley Lever Maggie Laubser Emilio Boggio Victor Stabin Giorgio De Vincenzi
Israel Tsvaygenbaum Arman Subhash Awchat Roger Ing Chen Ke (artist)

Training
Anton Mauve

Subjects (22)
People from Zundert · Post-impressionist painters · People of Montmartre · Royal Academy of Fine Arts (Antwerp) alumni · Académie Royale des Beaux-Arts alumni · Botanical illustrators · 1853 births · Dutch expatriates in Belgium · Suicides by firearm in France · Vincent van Gogh · Dutch expatriates in France · 1890 deaths · Dutch people with disabilities · Dutch Protestants · Male suicides · Dutch expatriates in the United Kingdom · Dutch Christians · Dutch male painters · Painters who committed suicide · Dutch still life painters · Flower artists

Figure 2: The home page of the developed web interface and an example of artist page.

not only a generic user but especially any art historian—directly with an easy-to-use exploration tool to view the properties of an artwork or an artist. An art historian, in fact, rarely analyzes artworks as isolated creations, but typically studies how different paintings, even from different periods, relate to each other, how artists from different countries and/or periods have exercised a influence on their works, how artworks completed in one place migrated to other places, and so on. The home page randomly loads artists and artworks. Each artist is associated with a page that reports information such as the biography, the works produced, etc. We leveraged the information provided by DBpedia to show also the fields, movements, other artists who have been influenced by the current artist, and many other tags. By clicking on the buttons, the user can browse the graph interactively. The page layout of an artwork is very similar to that of an artist and reports size, period, material, etc. It is also possible to browse the artworks according to the city/country in which they were completed or are currently located. When provided by

DBpedia, a textual description of the artwork is also shown.

The developed web interface can also show the results of some queries that can be particularly useful for art analysis, such as: retrieving the direct and indirect influencing connection between artists with different degrees of separation; identifying artworks that are stored in a country other than those in which they were completed; retrieving all the works that are are kept in a specific place; etc. On the tested platform, each query takes about a few tens of milliseconds. The ability to query the graph database already provides information retrieval and knowledge discovery capabilities in the art domain without having to train a learning system.

3. Multi-Task Multi-Modal Classification

ArtGraph encodes a valuable source of contextual knowledge to integrate with visual features automatically learned by deep neural networks to develop more powerful learning models in the art domain. Several tasks, in fact, could be addressed, such as artwork attribute prediction, multi-modal retrieval and artwork captioning, which are attracting increasing interest in this domain.

To this end, we propose a new classification model that is used in this paper to predict the artist, style and genre of a given artwork. The model is inspired by multi-modal learning: graph embeddings are extracted from *ArtGraph* using node2vec to provide the context information of the artwork; this information is intended to improve the accuracy of visual features extracted from the artwork using a pre-trained state-of-the-art CNN, i.e. ResNet50 [11]. The main idea is to learn how to project the visual features extracted by ResNet50 into the context space provided by the graph embeddings. This is done by an encoder module, consisting of two fully connected layers with a tahn activation function, so that values are between -1 and $+1$. The training phase focuses on minimizing the mean squared error (MSE) loss between the predicted embedding $\mathbf{p}_j$ and the true context embedding $\mathbf{u}_j$, for a given artwork instance j:

$$\ell_e(\mathbf{p}_j, \mathbf{u}_j) = \|\mathbf{p}_j - \mathbf{u}_j\|_2^2.$$

Then the predicted context features are combined (by concatenation) with the visual features. Instead of adding a single output layer and learning each classification task separately, we adopt a multi-task solution. In this way, features are shared between the tasks allowing the model to simultaneously exploit the semantic correlation between them to achieve better accuracy. Given a number of task T (three in our work, corresponding to the artist, style and genre classification) and a set of N instances, the overall loss function is:

$$\mathcal{L} = (1-\gamma)\left[\sum_{i=1}^{T} \lambda_i \sum_{j=1}^{N} \ell_c(\mathbf{z}_{ij}, class_{ij})\right] + \gamma \frac{1}{N} \sum_{j=1}^{N} \ell_e(\mathbf{p}_j, \mathbf{u}_j),$$

where γ weights the encoder module error, λ_i are hyperparameters that weight the contribution of each task i, ℓ_e is the aforementioned MSE loss and ℓ_c is the cross-entropy loss function defined as:

$$\ell_c(\mathbf{z}_j, class_j) = -\log\left(\frac{\exp(\mathbf{z}_j[class_j])}{\sum_i \exp(\mathbf{z}_j[i])}\right),$$

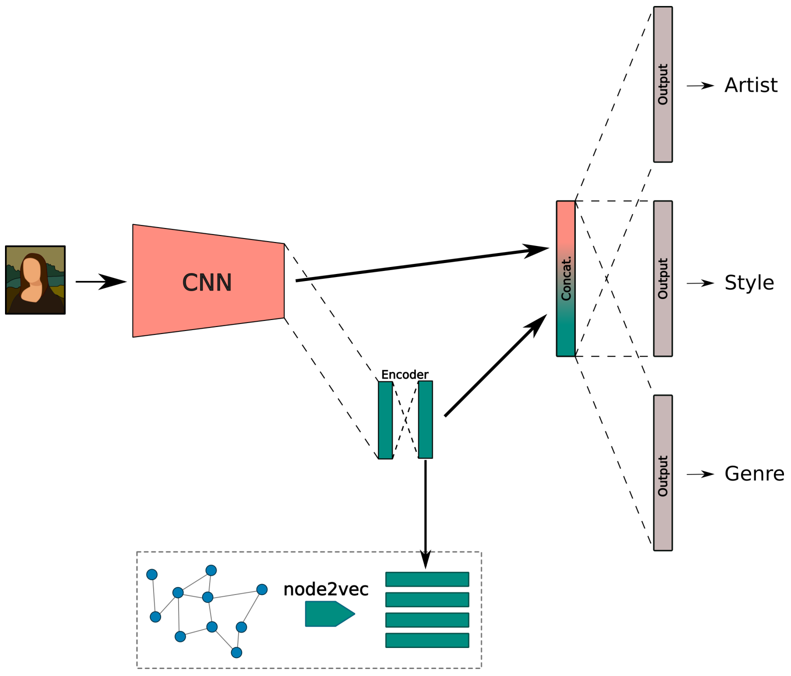


Figure 3: Proposed multi-task multi-modal model. A concatenation layer receives both the contribution of visual embeddings, extracted from ResNet50 trained on digitized images of the artworks, and graph embeddings extracted from our KG, respectively. The overall network learns to minimize the error made to predict the correct artist, style and genre of a given input and, at the same time, an MSE loss to project visual and contextual features in the same multidimensional space.

where, for a given artwork j, $\mathbf{z}_j$ is the predicted output and $class_j$ is the true label. An overall scheme of the proposed model is shown in Fig. 3.

The experiments were conducted on Google Colaboratory. The artwork images were resized to 224×224, as required by ResNet50, and normalized using the mean and standard deviation of ImageNet. The size of the visual embeddings produced by ResNet50 (without the output layer) is 2048, while the size chosen for the node2vec embeddings is 128. As an optimizer, we used Adam with learning rate 10^{-4} and momentum 0.9. The batch size was set to 32. In addition, we empirically found the following values for: γ, which was set to 0.4; λ_{artist}, set to 0.5; λ_{style}, set to 0.2; and λ_{genre}, set to 0.2. In other words, giving more importance to the classification loss and the artist contribution to this loss generally provides better performance.

It is worth noting that graph embeddings should not be learned on the entire graph, otherwise a bias would be introduced so that the model has already seen the test entities and their connections with the rest of the graph. Instead, we assume that at test time only the visual appearance of the artwork is known to the model, but the context information learned during training has already served to allow it to generalize beyond just the visual features. For this reason, we randomly divided our graph (and consequently the image set) into three sets: 80% for training, 10% for validation and 10% for test. The validation set was used to tweak the hyperparameters. Embeddings were only learned from the "training" graph.

The results obtained, expressed in terms of classification accuracy, are provided in Table 2.

Table 2
Results for the artist, style and genre classification tasks.

Method	Artist	Style	Genre
Fine-tuned ResNet	61.13%	62.65%	65.32%
ContextNet	62.20%	62.24%	65.93%
Proposed	**62.50%**	**65.93%**	**66.52%**

As a baseline for comparing our method, we experimented with a fine-tuned ResNet50, trained only on the digitized images. In addition, we compared our method with the *ContextNet* model proposed by Garcia et al. [8], which is also based on ResNet50 and uses graph embeddings only as a "regularization" signal but not as an additional input mode during training. We can see that models that incorporate contextual knowledge are better than the baseline method based only on visual features. Moreover, our model is able to better exploit context representation, with higher accuracy than *ContextNet* for all three tasks.

4. Conclusion & Future Work

In this paper, we have presented *ArtGraph*, an artistic knowledge graph primarily intended to provide art historians with a rich and easy-to-use tool to perform art analysis. This effort can foster the dialogue between computer scientists and humanists that is currently sometimes lacking [12]. Indeed, contrary to other works, we are not only interested in leveraging the KG information to learn classification tools, but also to help tackle knowledge discovery tasks. Humanists are interested not only in a classification model, but also in uncovering relationships, connections, trends and changes over the course of art history over time. Once stable, we will make *ArtGraph* publicly available to provide the pattern recognition and computer vision community with a good basis for further research on automatic art analysis.

As a future work, we want to tackle other significant tasks, such as multi-modal retrieval. Furthermore, we want to expand the proposed learning model by leveraging the Graph Convolutional Network framework, as recently done for example in [13].

Acknowledgments

G.V. acknowledges the financial support of the Italian Ministry of University and Research through the PON AIM 1852414 project.

References

[1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (CSUR) 54 (2021) 1–37.

[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (2015) 436–444.

[3] G. Castellano, G. Vessio, Deep learning approaches to pattern extraction and recognition in paintings and drawings: an overview, Neural Computing and Applications (2021) 1–20.

[4] E. Cetinic, T. Lipic, S. Grgic, Fine-tuning convolutional neural networks for fine art classification, Expert Systems with Applications 114 (2018) 107–118.

[5] C. Sandoval, E. Pirogova, M. Lech, Two-stage deep learning approach to the classification of fine-art paintings, IEEE Access 7 (2019) 41770–41781.

[6] G. Strezoski, M. Worring, Omniart: a large-scale artistic benchmark, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14 (2018) 1–21.

[7] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.

[8] N. Garcia, B. Renoust, Y. Nakashima, ContextNet: Representation and exploration for painting classification and retrieval in context, International Journal of Multimedia Information Retrieval 9 (2020) 17–30.

[9] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: ACM SIGKDD, 2016, pp. 855–864.

[10] N. Garcia, G. Vogiatzis, How to read paintings: semantic art understanding with multimodal retrieval, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018.

[11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[12] G. Mercuriali, Digital art history and the computational imagination, Int J Digit Art Hist Issue 3 2018 Digit Space Architect 3 (2019) 141.

[13] C. B. E. Vaigh, N. Garcia, B. Renoust, C. Chu, Y. Nakashima, H. Nagahara, GCNBoost: Artwork classification by label propagation through a knowledge graph, arXiv preprint arXiv:2105.11852 (2021).

Generating Table Vector Representations

Aneta Koleva[1,2], Martin Ringsquandl[1], Mitchell Joblin[1] and Volker Tresp[1,2]

[1]*Siemens, Otto-Hahn-Ring 6, 81739 Munich, Germany*

[2]*Ludwig Maximilian University of Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany*

Abstract

High-quality Web tables are rich sources of information that can be used to populate Knowledge Graphs (KG). The focus of this paper is an evaluation of methods for table-to-class annotation, which is a sub-task of Table Interpretation (TI). We provide a formal definition for table classification as a machine learning task. We propose an experimental setup and we evaluate 5 fundamentally different approaches to find the best method for generating vector table representations. Our findings indicate that although transfer learning methods achieve high F1 score on the table classification task, dedicated table encoding models are a promising direction as they appear to capture richer semantics.

Keywords

table interpretation, table classification, representation learning.

1. Introduction

Tabular data is one of the most prevalent data representations. The effort by Cafarella [1], known as WebTables, identified and extracted more than 200 million high-quality tables from HTML pages. The availability of such large corpus of structured data initiated several directions of research related to the different applications of tabular data such as: table search [2], table improvement [3], question answering [4], and semantic annotation of columns [5]. As a result of the increasing adoption of KGs, which are often populated from tabular data, the task of aligning tables with KGs, also referred to as table interpretation (TI), has become a highly relevant task. In contrast to information extraction from unstructured documents, TI should leverage the explicit relational structure. The unique table structure with rows and columns of cells and other *metadata* can be exploited for discovery and disambiguation of the meaning captured in the table. The task of TI entails three different sub-tasks. The first sub-task, which is the focus in this paper, is the classification of tables according to classes in a given KG schema. The second sub-task is related to linking rows from tables to existing entities in the KG. The annotation of columns as entity attributes and the discovery of binary relations between columns is the third sub-task of TI. While there have been several works focusing on the row-to-entity [6, 7, 8], and column-to-attribute sub-tasks [5, 9], the task of linking a table to a class has been neglected. However, in the case of *entity tables*, where one column (the *core column*) is associated to the name of the entity and the remaining columns are attributes of this entity, discovering the class of the table as a first step can greatly improve the solving of the other two sub-tasks. It is often the case that the column names are missing or incorrect, therefore finding the name of the core

ISWC 2021 Workshop DL4KG

firstname.lastname@siemens.com

column does not imply finding the class of the table. Moreover, when two tables have the same column names and similar content (e.g., one table of class *Country* and one of class *City*), it is not trivial to disambiguate the entities and column types based only on the table content. Once a table has been *interpreted*, its content can be used for extracting new triples for enriching the KG, a task known as *KG completion*, or for extracting missing facts for the KG, which is the task of *slot-filling*.

Due to the inherent scarcity of labelled data for the first sub-task (class-annotated tables), a table classification model must either be of low complexity (few parameters) or leverage pre-trained models. Using pre-trained models in TI has been studied only to a very limited extend. Hence, we explore two promising directions for making learning-based approaches more efficient: (a) by using transfer learning, (b) by considering additional inductive biases that are unique to tabular data representations.

We propose an experimental setup with the intention of finding the best method for generating a representation which captures the information from the table but also the row and column structure, so that it can be later used towards solving the remaining sub-tasks of TI: row-to-entity linking, column type annotation and relation extraction. We are interested in understanding how pre-trained language models, such as BERT [10], and their dedicated table-based counterparts, for instance TaBERT [11], can be utilized for generating vector representation for table. Surprisingly, our experiments show that a transfer learning method with a rich vocabulary of pre-trained word embeddings achieves similar F1 score compared to more sophisticated pre-trained language models (LM). Another interesting finding is that the inductive bias for tabular structure in the LM pre-trained on tabular data does not bring beneficial impact to a text pre-trained LM. However, the classification confusion matrix for this method, gives an insight to the miss-classifications being justifiable and reasonable. Our main contributions are:

- A formal definition of table classification as a machine learning task and a protocol for evaluating performance on this task.
- A setup for table encoding using 5 fundamentally different approaches covering a spectrum of paradigms from general purpose document encoders to specialized pre-trained models designed for tabular data.
- An extensive empirical evaluation of the different approaches.

2. Background

In this section, we review prior work related to solving the different sub-tasks of TI. We also give a short overview of methods for generating vector representations of tables.

Table Interpretation The three sub-tasks of TI were first introduced in the paper by Ritze et al. [12]. That paper also introduced the T2K Matcher, a method for iterative value-based matching, which solves the TI tasks by matching values from the tables to values of retrieved candidates from the KG. More recent work by Limaye et al. [9] proposed a probabilistic graphical method which attempts to jointly solve the two sub-tasks of finding entity-to-row and column-to-attribute alignments. Deng et al. [13] exploited word embeddings for representing the contents of tables and utilized them for the discovery of new entities. The SemTab challenge

[14] has also motivated new approaches [15, 16]. However, the task of table-to-class annotation is not part of this challenge.

Table classification To the best of our knowledge, the T2K Matcher is the only existing method for solving the table-to-class task. Namely, the class of the table is chosen by ranking the sum of the similarity scores of the column-to-property correspondences aggregated per class. Since this method requires querying of the KG for candidate retrieval and first solving the column-to-property alignment in order to find the correct class of a table, we do not consider it during our experiments. In contrast to the T2K Matcher, we consider a *closed book* scenario, where the instances of the KG are not available, only the classes in the KG schema.

Representation Learning on Tables Based on powerful LM, dedicated *deep learning* models have recently been proposed to exploit tabular data structures, e.g., in table-based question answering [4, 17] and KG completion from tables [18]. One benefit from using pre-trained LM is that they can handle synonyms well, e.g., the abbreviation of New York as NY, which are frequently occurring in tables because of the innate limitation of the cells. The other benefit is that, due to the exposure to large textual corpora during the pre-training phase, the LM can *store* implicit information learned from the data whilst pre-training, in the form of model parameters [19]. TaBERT [11] by Yin et al. is a novel model which was pre-trained to jointly learn representation of a natural language question, called *utterance*, and tables. An example of utterance for the entity table shown in Figure 1 is the question: *How much is the population of New York?*. During encoding, instead of using the full table, TaBERT samples 1 or 3 rows, referred to as *content snapshot*. First, each row from the snapshot, concatenated with the utterance, is encoded by BERT [10]. Second, the encoding of the rows are stacked and in order to generate vector representations for each of the columns, a vertical self-attention mechanism is used. Finally, representation for the table is generated by pooling the column representations. Similar work is the method TAPAS by Herzig et al. [20], which is also pre-trained on tables and text segments. Ding et al. proposed TURL [17] as a framework for pre-training, also on tabular data, which uses the same objectives as TaBERT for learning representations of the content of the tables. Additionally, they proposed task-specific fine-tuning on the framework for solving the row-to-entity and column-to-attribute annotation. Wang et al. [21] presented a novel method which exploits information within one table but also aggregates the contextual information shared across similar tables in order to generate a vector representation that can be used for column-to-class annotation and relation prediction tasks.

3. Problem Description

We focus on the task of table-to-class annotation. The task has been introduced together with the two other TI sub-tasks in [12], however without a formal definition. The goal of the table-to-class annotation is to label a table with its corresponding class according to the given KG schema. We now provide a definition of this task as a machine learning task.

An entity table $\boldsymbol{T_i}$ is a $N_i \times M_i$ matrix where N_i and M_i are the number of rows and columns of the table $\boldsymbol{T_i}$. Each element of the matrix $\boldsymbol{T_i}$, $r^i_{n,m}$, contains one or more tokens, where each token is a sequence of characters. We denote with $r^i_{n,*}$ and $r^i_{*,m}$ the n-th row and the m-th column of the matrix $\boldsymbol{T_i}$ respectively. The header of the table is the first row $H_i = r^i_{0,*}$. The content of the table are the rows $r^i_{1,*}, r^i_{2,*}, \ldots, r^i_{N,*}$.

Let $\mathscr{D} = \{(\boldsymbol{T}_1, c_i), \ldots, (\boldsymbol{T}_l, c_i)\}$ be the set of labeled tables with l number of tables, and each label $c_i \in C$ is in the set of classes defined in the KG schema $\mathscr{C} = \{c_1, \ldots, c_k\}$. A table encoder $E_{\boldsymbol{\omega}}$ is a model, with a parameter vector $\boldsymbol{\omega}$, which encodes each table $E_{\boldsymbol{\omega}} : \{\boldsymbol{T_i}\} \rightarrow \mathbb{R}^d$ to a vector $E_{\boldsymbol{\omega}}(\boldsymbol{T_i}) = \boldsymbol{x_i}$ and $\mathscr{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$ is the set of feature vectors for every $\boldsymbol{T_i} \in \mathscr{D}$. The final task is to train a classification model $f_{\boldsymbol{\theta}} : \mathbb{R}^d \rightarrow \mathscr{C}$ so that each table vector is assigned to one of the class labels. The problem is defined in the multi-class setting. Formally our setting is $f_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\omega}} : \{\boldsymbol{T_i}\} \rightarrow \mathscr{C}$, where only the parameters $\boldsymbol{\theta}$ are trained on the table classification task, i.e., no gradient updates are performed on $\boldsymbol{\omega}$.

4. Experiments

Figure 1 shows the experimental setup for evaluating different table encoders. Given an entity table, a table encoder generates a high-dimensional vector representation of the table. We then train a classifier on the table-to-class task and evaluate the performance achieved by each of the table encoders. We experiment with different types of table encoders, a simple method such as document encoder, transfer learning methods with general-purpose pre-trained word embeddings (Figure 1 (a)) and more complex methods which include a LM pre-trained on large textual corpora and an approach for question-answering which has been pre-trained on tabular data (Figure 1 (b)). The code for the experiments is accessible online [1].

4.1. Dataset

For evaluation we used the second version of the T2D gold standard dataset [12], T2Dv2. To the best of our knowledge, the T2D sets are the only publicly available datasets which have been annotated with table-to-class correspondence. The second version of the dataset[2] contains 237 such annotations. In our experiments, we consider those classes which have at least two tables as representatives. The resulting dataset contains 223 tables, each labeled with one of the 27 unique classes. The mean of the number of rows in the dataset is 119.2 and the mean of the number of columns is 7.7.

4.2. Models compared

In the evaluation we used 5 different models as table encoders, varying from general purpose document encoders to more sophisticated LM, pre-trained on tabular data.

TF-IDF or term frequency-inverse document frequency, is a term weighting scheme which generates vector representation for a document based on the frequency of the words in the document. It is the simplest method which we used as a table encoder.

[1]https://github.com/anetakoleva/tableClassification
[2]http://webdatacommons.org/webtables/goldstandardV2.html

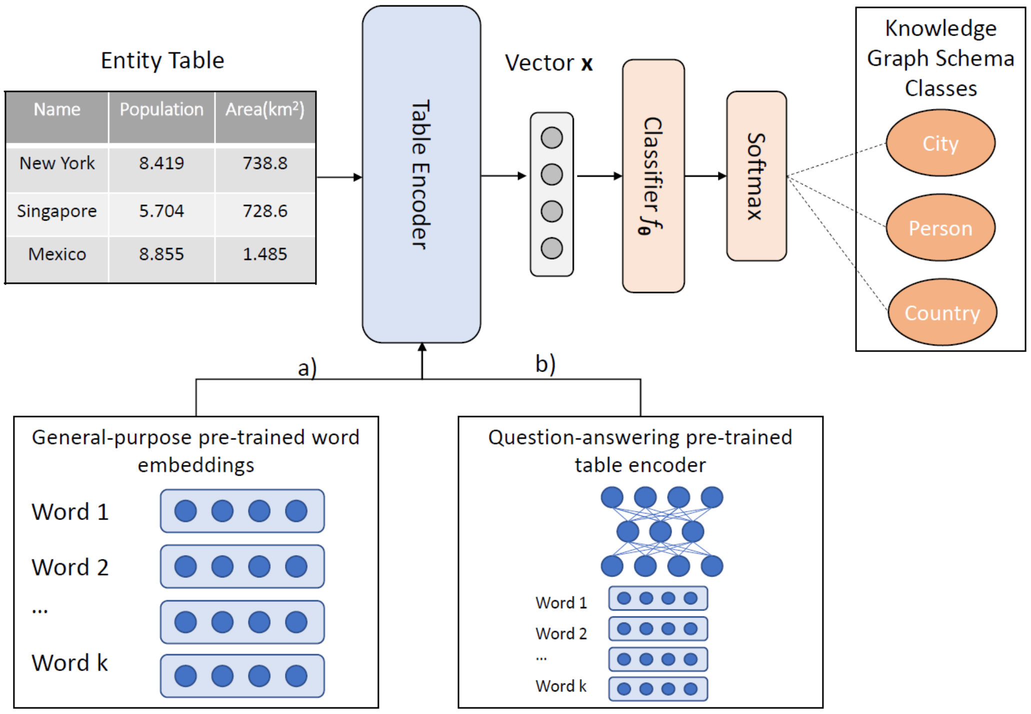


Figure 1: Experimental setup for evaluation of table encoders.

Spacy pre-trained word vectors on a text extracted from blogs, news and comments. We used the vectorizer from english-medium sized pipeline[3] which contains vocabulary of size 684830.

Word2Vec pre-trained word vectors trained with FastText [4] on a Wikipedia text corpus. The model used for the learning the vectors [22] is an extension of the original word2vec model. It is skip-gram based and trained to learn representations for character n-grams. This model consists of vocabulary of size 2.5 million.

BERT is a widely used, Transformer-based LM [10]. During the pre-training phase, the model has been exposed to a large corpus of unstructured text with the objective of predicting missing words and prediction of next sentence. This enables the model to learn the correlation of the words and to generate different vector representation for words depending on the context.

TaBERT is a table encoding method [11], pre-trained on Web tables with the objective to be used in question-answering tasks on tables. Since the model expects an *utterance*, i.e., a natural language question, as input together with a table, in our experiments we provided an empty space " ". We conducted more experiments to evaluate the influence of the utterance on the generated table representation and we discuss these results in Section 5.

[3]https://spacy.io/models/en#en_core_web_md
[4]https://fasttext.cc/docs/en/pretrained-vectors.html

4.3. Setup

To systematically evaluate the quality of the representations generated with the different table encoders, we compare their performance on the classification task under different scenarios. It is important to note that we did not train or fine-tune any of the methods for table encoding, i.e., we used them *off-the-shelf*. Since the tables can be large, in order to avoid scalability issues, we resort to sampling of rows. Namely, we first shuffle the rows in the tables and then we sample the first q rows. The shuffling of the rows is done only once. For the experiments, we sampled $q \in \{1, 3, 5, 7\}$ rows from each of the tables and used these sampled tables as input to the table encoders.

When using TF-IDF as table encoder, the input is a set of sequences, where each sequence corresponds to a table from the set of tables $\mathscr{D}$. More formally, a table sequence for table $\boldsymbol{T_i}$ is a sequence of rows $S_{T_i} = (r^i_{0,*}, r^i_{1,*}, \dots, r^i_{q,*})$, such that $q \in \{1, 3, 5, 7\}$, and the set of sequences is the set $I = \{S_{T_0}, \dots, S_{T_l}\}$. The table encoder TF-IDF transforms the set of table sequences to the set of feature vectors $E^{\text{tf-idf}}_{\omega} : I \rightarrow \mathscr{X}$.

Word2Vec and Spacy generate the vector representation for table $\boldsymbol{T_i}$ in 3 steps. First, the sequence S_{H_i}, representing the header of the table T_i, is encoded as the mean over the word vectors in the sequence S_{H_i}, represented as $\boldsymbol{x}^i_H$. Second, the content of the table, is transformed into a table sequence $S_{T_i} = (r^i_{1,*} \dots r^i_{q,*})$ and encoded as the vector $\boldsymbol{x}^i_B$, which represents the mean over all the word vectors in S_{T_i}. Finally, the vector representations for the header and for the table content are concatenated into one vector $\boldsymbol{x}_{T_i} = \boldsymbol{x}^i_H \| \boldsymbol{x}^i_B$.

Considering that there is a limit on the length of the sequence that BERT can encode in one step, we used different transformation for the last two methods. BERT encodes each table row by row, i.e, a sequence $S^i_{r_{z,*}}$ is generated for each of the rows $r^i_{z,*}$ of table $\boldsymbol{T_i}$, where $0 \leq z \leq q$. BERT generates row-wise vectors, so for each sequence $S^i_{r_{z,*}}$ the output is a vector $\boldsymbol{x}_{r_{z,*}}$. The vector representation for table $\boldsymbol{T_i}$ is the vector $\boldsymbol{x}_{T_i}$ which is the result of the mean-pooling over the set of the BERT's output vectors $\{\boldsymbol{x}_{r_{0,*}}, \dots, \boldsymbol{x}_{r_{q,*}}\}$ that correspond to the table rows. In the same manner, the TaBERT model also first generates an encoding for each of the rows of table $\boldsymbol{T_i}$ resulting in a set of vectors. This model uses vertical self-attention focused on the vertically stacked vectors, $\{\boldsymbol{x}_{r_{0,*}}, \dots, \boldsymbol{x}_{r_{q,*}}\}$. Because of the vertically aligned vectors, the output of the model is a column vector representation $\{\boldsymbol{x}_{r_{*,0}}, \dots, \boldsymbol{x}_{r_{*,M_i}}\}$ for each of the M_i columns in table $\boldsymbol{T_i}$. Finally, we do mean-pooling over the column representations to generate the table encoding $\boldsymbol{x}_{T_i}$.

We then use the Multi-layer Perceptron (MLP) with one hidden layer of size 500, the *tanh* activation function and *adam* optimizer as the classifier $f_{\boldsymbol{\theta}}$ from Figure 1. The hyper parameters are chosen after an extensive search and they are fixed for all of the experiments. Since the available dataset is small, instead of splitting it once into a training set and a test set, we use stratified K-fold validation with $K = 20$ splits. Considering that the dataset is imbalanced, we report the macro averaged F1 score. The reported scores are the average of the results on the test set after the cross validation. To explore the effect of the column names, we also encoded the tables with their column names masked. Specifically, for all of the tables, we substitute their column names with the token [UNK].

Table 1
Macro-averaged F1 score.

	Column names				**Masked column names**			
	$q = 1$	$q = 3$	$q = 5$	$q = 7$	$q = 1$	$q = 3$	$q = 5$	$q = 7$
tf-idf	0.56	0.56	0.54	0.54	0.41	0.45	0.51	0.55
spacy	0.64	0.69	0.74	0.73	0.48	0.58	0.61	0.63
word2vec	0.69	0.76	0.76	0.78	0.61	**0.77**	0.76	**0.80**
bert	**0.76**	**0.78**	**0.79**	**0.80**	**0.63**	0.75	**0.78**	0.78
tabert	0.75	0.77	0.77	0.78	0.61	0.71	0.71	0.74

5. Results

Table 1 shows the macro averaged F1 score for the 5 table encoders on the table classification task under two different settings: (1) given the input tables with the column names and (2) given the input tables with their column names masked ([UNK] token). We report the achieved F1 score for the different sizes of the input tables with the number of sampled rows q varying from 1 row to 7 rows. The simplest table encoder, TF-IDF achieves the lowest F1 score and the score only got lower when the column names of the tables were masked. For the two models with pre-trained word vectors, we observe that the model with the richer vocabulary has higher score. Indeed, the F1 score of Word2Vec is comparable with the scores achieved by BERT and TaBERT. In the first setting, when the column names of the tables are visible, there is no significant difference between the scores achieved by BERT and the scores of TaBERT. However, in the setting when the column names are masked, BERT consistently outperforms TaBERT. Interestingly, Word2Vec is the only table encoder that was not affected by the masking of the column names, on the contrary, it achieved better score in the case when $q = 3$ and $q = 7$ under the second setting compared to the setting when the column names are visible.

Figure 2 shows the row-normalized confusion matrix for the table classification task for Word2Vec and TaBERT across the different classes. The horizontal axis shows the predicted labels and the vertical axis shows the true labels. We observe the performance of the two models under the same scenario: the input tables are with $q = 7$ rows and the column names are masked. The classes are ordered by the number of instances assigned to them, *Country* is the class with the most instances, 33, while *Airline* has only 2 instances. From the confusion matrix for TaBERT (Figure 2 right) it can be observed that more miss-classifications are for the classes with a lower number of instances and they are not that unexpected. For instance, miss-classifying an instance of class *Person* as an instance of class *Scientist*, is an acceptable mistake. Similarly for the instances of classes *Academic Journal* and *Newspaper*, and *Political Party* and *Election*. On the other hand, the miss-classifications by Word2Vec for class *Wrestler* and class *Animal* as instances of class *Film* are much more unexpected and critical. Likewise, Word2Vec miss-classifies the tables of class *Scientist* and of class *Radio Station* as instances of the class *Country* which indicates a weak semantic structure in the vector representations. These results suggest that although Word2Vec achieves higher F1 score, the TaBERT vector representations capture semantics with a smoother transitions between classes.

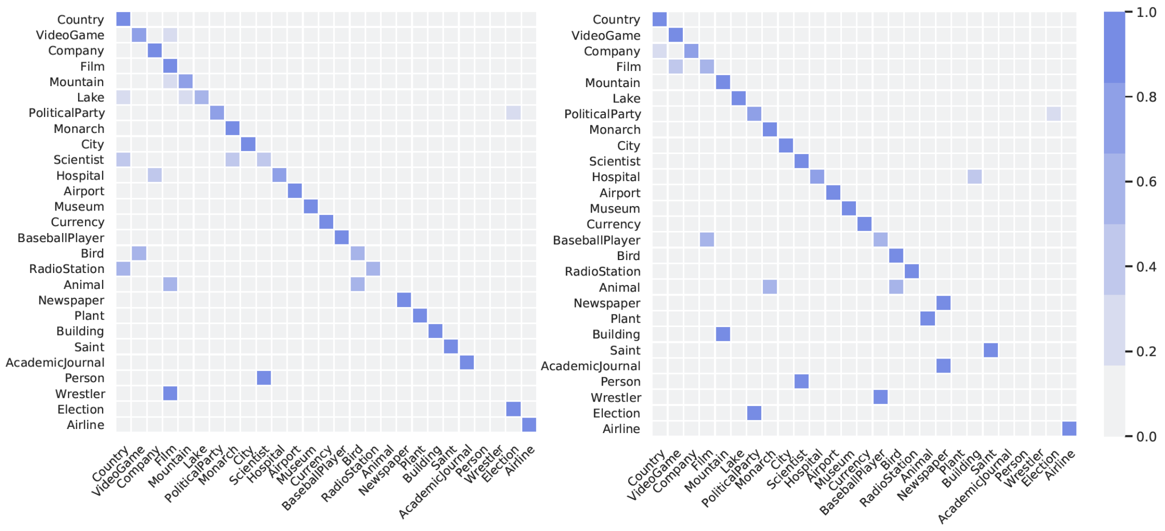


Figure 2: Classification confusion matrix for Word2Vec (left) and for TaBERT (right).

TaBERT Analysis To get a better understanding of the (under-) performance of TaBERT we analyse the influence of the *utterance* and its interplay with column names. In addition to the empty string “ ” used in previous experiments, we also used a randomly generated string with 10 characters (unique per table), and one constant string, *Thing*, for all tables. Moreover, we experimented with adding the correct class of the tables as utterance, as well as a wrong class (for instance, all the tables of class *Country* are encoded with the class *Plant* as utterance). Figure 3 shows the results of these experiments, where the input tables were with $q = 3$ rows. The horizontal axis shows the different options that we passed as utterance to the model and the vertical axis shows the achieved F1 score. The masking of column names has significant influence on the generated table representation. The reason for this might be in the way how a row is transformed into a string, i.e., the value of each table entry is concatenated with the column name of the entry and its value. Observing the results with the different utterance, we see that the choice of utterance does not affect the performance of the model when the column names are not masked. Nevertheless, when the column names are masked, the influence of the utterance is more significant. In both cases when the utterance is the wrong class or the correct class, the achieved score is much higher, which might be attributed to a class-wide shift in the vector space because of the grouping that these utterances cause.

6. Conclusion and Future work

In this paper we explored different types of table encoders for generating vector representations for tabular data. Specifically, we focused on evaluating different methods for table encoding on the sub-task for TI, table-to-class annotation. Despite the increasing interest in the problem of TI, so far, only one approach towards this specific sub-task has been proposed. In this direction, we provided a formal definition for the table-to-class annotation task as a machine learning task. We conduct an empirical study with five different methods for generating vector

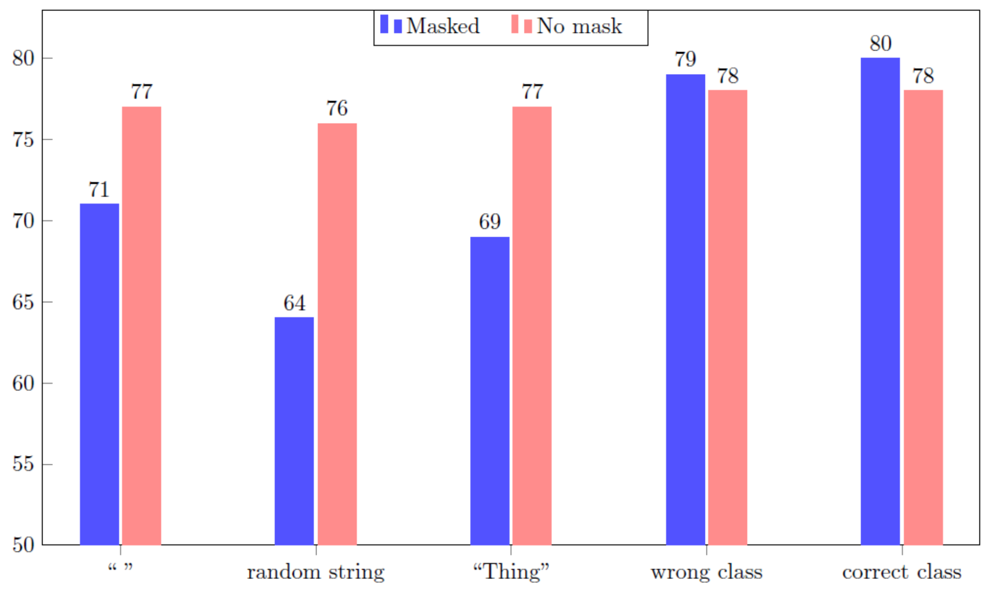


Figure 3: TaBERT performance with different utterances.

representation of a table and evaluate their performance on the table-to-class annotation task. The results from our experiments show that transfer learning methods with large vocabularies of pre-trained word embeddings perform on par with more complex and expensive modes such as LM pre-trained on tables. An interesting finding is that the inductive bias for tabular structure in TaBERT did not bring benefit to the performance of the BERT model. A possible explanation for this is the missing significant utterance that the TaBERT model expects as input. Nonetheless, the miss-classifications made by this model are reasonable, suggesting that the vector representations capture the semantics of the tables. Future work should target closing the gap between existing general-purpose models and model specific for encoding tabular data. To further our work we plan to explore other existing methods for table encoding for solving the table-to-class task, as well as for solving the entity-to-row and column-to-property tasks.

References

[1] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, Y. Zhang, Webtables: exploring the power of tables on the web, VLDB (2008).

[2] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu, Recovering semantics of tables on the web, VLDB (2011).

[3] M. Zhang, K. Chakrabarti, Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables, in: SIGMOD, 2013.

[4] H. Sun, H. Ma, X. He, W. Yih, Y. Su, X. Yan, Table cell search for question answering, in: WWW, 2016.

[5] J. Chen, E. Jiménez-Ruiz, I. Horrocks, C. Sutton, Learning semantic annotations for tabular

data, in: IJCAI, 2019.

[6] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, V. Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: ISWC, 2017.

[7] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, Tabeano: Table to knowledge graph entity annotation, CoRR (2020). `arXiv:2010.01829`.

[8] S. Zhang, E. Meij, K. Balog, R. Reinanda, Novel entity discovery from web tables, in: WWW, 2020.

[9] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, VLDB (2010).

[10] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, 2019.

[11] P. Yin, G. Neubig, W. Yih, S. Riedel, Tabert: Pretraining for joint understanding of textual and tabular data, in: ACL, 2020.

[12] D. Ritze, O. Lehmberg, C. Bizer, Matching HTML tables to dbpedia, in: WIMS, 2015.

[13] L. Zhang, S. Zhang, K. Balog, Table2vec: Neural word and entity embeddings for table population and retrieval, in: SIGIR, 2019.

[14] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: ESWC, 2020.

[15] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J. Yao, J. Williams, A. Gordon, C. Lin, Linkingpark: An integrated approach for semantic table interpretation, in: SemTab@ISWC, 2020.

[16] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata, in: SemTab@ISWC, 2020.

[17] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, TURL: table understanding through representation learning, VLDB (2020).

[18] B. Kruit, P. A. Boncz, J. Urbani, Extracting novel facts from tables for knowledge graph completion, in: ISWC, 2019.

[19] A. Roberts, C. Raffel, N. Shazeer, How much knowledge can you pack into the parameters of a language model?, in: EMNLP, 2020.

[20] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, J. M. Eisenschlos, Tapas: Weakly supervised table parsing via pre-training, in: ACL, 2020.

[21] D. Wang, P. Shiralkar, C. Lockard, B. Huang, X. L. Dong, M. Jiang, TCN: table convolutional network for web table interpretation, in: WWW, 2021.

[22] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics (2017).

Understanding Class Representations: An Intrinsic Evaluation of Zero-Shot Text Classification

Fabian Hoppe[1,2], Danilo Dessì[1,2] and Harald Sack[1,2]

[1]*FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany*
[2]*Karlsruhe Institute of Technology, Institute AIFB, Germany*

Abstract
Frequently, Text Classification is limited by insufficient training data. This problem is addressed by Zero-Shot Classification through the inclusion of external class definitions and then exploiting the relations between classes seen during training and unseen classes (Zero-shot). However, it requires a class embedding space capable of accurately representing the semantic relatedness between classes. This work defines an intrinsic evaluation based on greater-than constraints to provide a better understanding of this relatedness. The results imply that textual embeddings are able to capture more semantics than Knowledge Graph embeddings, but combining both modalities yields the best performance.

Keywords
Zero-Shot Learning, Text Classification, Class Representation, Embedding Model, Intrinsic Evaluation

1. Introduction

Managing, finding and exploring information from textual data is frequently done by applying Text Classification. As such classifying texts according to a predefined taxonomy is a key task in Natural Language Processing and Information Retrieval. For example within the scholarly domain classification supports researchers to retrieve relevant articles for their research by categorizing huge document collections according to a given schema. In order to achieve this goal Text Classification requires a certain understanding of the information presented in natural language texts. Typically, supervised classifiers obtain this understanding by statistically analyzing features of large training sets. In the last decade, the required amount of task-specific training data was to some extent reduced by pre-training large language models on the cloze task or similar self-supervised tasks on large unlabeled corpora. Nevertheless, this still requires a sufficient number of training examples for each class aside from the taxonomy itself. Moreover, collecting training data is costly, because human experts have to label each document and, therefore, this time consuming process is not feasible for many classification tasks. Classes might follow a long-tail distribution, i.e., there are many classes with only a few examples, or the taxonomy might get frequently extended by emerging new classes. One way of coping with insufficient training data is to exploit external knowledge about given classes, mimicking how

Workshop on Deep Learning for Knowledge Graphs (DL4KG@ISWC2021), October 25, 2021
fabian.hoppe@fiz-karlsruhe.de (F. Hoppe); danilo.dessi@fiz-karlsruhe.de (D. Dessì); harald.sack@fiz-karlsruhe.de (H. Sack)
0000-0002-7047-2770 (F. Hoppe); 0000-0003-3843-3285 (D. Dessì); 0000-0001-7069-9804 (H. Sack)

humans learn to classify documents; this is what Zero-shot Text Classification aims to achieve. More precisely, instead of classifying documents by comparing them to other documents of a specific class, Zero-shot Classification exploits known relations between classes. Consequently, it does not require training data for all classes of a taxonomy and makes it possible to predict classes, which are not seen during training. This is achieved by transferring information from known classes to classes that are not seen or are not sufficiently represented. Practically, Zero-shot Classification aligns a class embedding space generated by using external knowledge with a document embedding space, and uses the interrelation between seen and unseen classes in their embedding space to obtain a representation for the unseen class in the aligned vector space. The actual classification is performed by applying a distance metric to find the classes most similar to the given documents. By definition the performance of such a classifier relies heavily on the utilized external knowledge of the classes and how it is represented in the class embedding space.

Many models focus on textual data as external knowledge and uses language models to create a vector space for these classes [1, 2]. More recently, efforts are made to include explicit knowledge by utilizing knowledge graphs, such as ConceptNet [3] or DBpedia [4]. However, improving the state of the art by providing the best suitable external knowledge using the best suitable embedding model requires a better understanding on how well the considered external knowledge is actually encoded in the embedding space. Certainly, this understanding is difficult to obtain by using extrinsic evaluation of the whole model, because these results are heavily influenced by other factors, e.g., the used document representations as well as the model used to perform the alignment of both spaces. Consequently, a sound judgment of the usefulness of the class representations on their own requires a more detailed review. Therefore, this paper proposes an intrinsic evaluation which investigates the class embedding space independently. The main requirement of class representations is an accurate encoding of the interrelations between classes, because these relations are exploited by Zero-shot Classification. It means that related or similar classes should be represented close to each other and unrelated or dissimilar classes should be further apart. Based on this intrinsic evaluation the most common external knowledge sources and the related embedding models are investigated on the example of the arXiv category taxonomy[1]. This investigation uses a newly created gold standard which is made publicly available together with the source code for the detailed investigation of class representations[2]. Thereby, this paper aims to facilitate further research into the used external knowledge for class representations and the applied embedding models and improve Zero-shot Text Classification.

In summary, the contribution of this work is threefold:

- An evaluation of class embeddings for Zero-shot Classification is proposed.
- A new gold standard based on the arXiv category taxonomy is provided.
- Common class representations are evaluated and insights to obtain better classification results are discussed.

The reminder of this paper is organized as follows. Section 2 discusses the relevant related work in Zero-shot Text Classification and the evaluation of representations. Afterwards, Section 3

[1]https://arxiv.org/category_taxonomy
[2]https://github.com/ISE-FIZKarlsruhe/IntrinsicEvaluationOfClassRepresentations

introduces common external knowledge sources and embedding models and describes the intrinsic evaluation. Section 4 reports and discusses the results of this intrinsic evaluation on arXiv class representations. Finally, Section 5 concludes the paper and outlines future directions.

2. Related Work

One of the first works exploring Zero-shot Text Classification is the dataless classification framework [5]. The authors use Explicit Semantic Analysis as a latent representation for documents and classes, which avoids the alignment of both spaces. The class representations are generated based on the class names as external knowledge resource. This research got extended by considering several neural network based word embeddings, such as Word2Vec [1]. Due to the omitted alignment step these approaches do not require any training data. Consequently, similar models are deployed, where no training data is available, like categorizing German archival documents [6]. Recently, contextualized embedding models are utilized by reformulating the classification task as a textual entailment problem [2]. This approach continues to represent classes by only using the class name in combination with large, self-supervised language models as external knowledge. However, as shown by [7] for Zero-shot Image Classification already a basic classifier using a linear transformation to align image and GloVe class embeddings can reach state-of-the-art performance by considering Wikipedia articles as additional external knowledge for the given classes. Recent studies in Zero-shot Text Classification also extend their model by including explicit knowledge sources, like Knowledge Graphs (KGs), to generate suitable class representations. For example, in [3] the ConceptNet KG is used to extract explicit relations between words. Another work investigates the usage of DBpedia RDF2Vec embeddings to represent arXiv categories [4]. The authors investigate whether text embedding combined with KG embeddings would provide better class representations for a Zero-shot Classification.

Regardless of the increased focus on class representations state-of-the-art Zero-shot models still rely on costly extrinsic evaluations to judge these representations, i.e., the quality of embedding models is assessed by only considering the performance of the task itself. Thus, it requires more computational power and is not suited to provide a better understanding on the representations. Other tasks utilizing embedding models apply intrinsic evaluation to gain a better understanding of the applied embedding model. Instead of considering the whole system, these evaluations focus on specific intermediate subtasks. Due to their wide-spread use especially the semantics of word embeddings is extensively investigated based on multiple intrinsic evaluations [8]. The most popular are word semantic similarity and the word analogy task. The word semantic similarity task evaluates the correlation between similarities of embedding pairs to human labelled semantic similarity. This provides a direct evaluation of the relations between words. However, the task is quite subjective and depends on the relations considered for the human labels, e.g. a football and a globe are both spheres and would be similar if the shape property is most important for the given task. The word analogy task (a is to $\hat{a}$ as b is to $\hat{b}$) tries to reduce this influence by considering only a given relation which is defined by the first pair $(a, \hat{a})$. Unfortunately, it requires a careful selection of word pairs so that the relation makes sense which increases the labelling effort. After all, both tasks provide a better understanding of the created embedding space by direct evaluation of semantic relatedness. The importance of such an understanding is recently highlighted in [9]. The detailed investigation of KG embeddings by means of clustering and classification experiments raised doubt about the semantic capabilities of common used models. Nevertheless, the literature describes several pitfalls of intrinsic evaluation that need to be considered. Most commonly (i) it might fail to relate intrinsic and extrinsic evaluation [10], (ii) human annotators might introduce biases based on their background [11], (iii) low inter-annotator agreement [12]. However, intrinsic evaluation is crucial to understand the models and to know what to improve.

3. Class Representations

Class representations play a vital role for Zero-shot Text Classification. They encode external knowledge which is exploited to accomplish the classification of unseen or not sufficiently represented classes. Therefore, the understanding of the relevant external knowledge and the related embedding models is important. Before discussing how intrinsic evaluation provides a better understanding of these class representations, the most common models are briefly presented. Currently, two modalities of external knowledge are considered for the generation of class representations in Zero-shot Text Classification: textual data and Knowledge Graph-based data. Both modalities use their own embedding models.

3.1. Textual Representations

Typically, classes are described with natural language text to support the annotation by human experts as well as provide an easily understandable definition of these classes to the users. Therefore, this external knowledge is frequently available without any additional effort. Considering that this data informally defines the classes for the users it ranges from names of common, well-known classes to more detailed descriptions of specific classes depending on the classification task. In addition to these commonly available texts, classes can be linked to resources providing more background knowledge. If these links are not already part of the taxonomy, they can be retrieved by manual or (semi-)automatic entity-linking using the available texts. Even for large taxonomies this additional effort is dwarfed by the task of providing thousands of training examples for each class. One commonly used external resource is Wikipedia. As the largest encyclopedia it provides background knowledge for many classes in a broad set of domains.

The textual data is embedded into a vector space by using pre-trained language models. These models learn latent representations of words based on a word prediction task. As such they encode the usage of each word into a vector space. Words that frequently occur in the same context are represented as similar vectors. **Word2Vec** [13] is one example of such an embedding model. It provides two settings. The *skip gram* setting computes the embedding of a word given its surrounding context, whereas the *CBOW* setting computes the embeddings of a context, given a word. In the Word2Vec model a word is always represented by one single vector. This poses two challenges. First, a longer text sequence requires additional normalization steps and secondly, the ambiguity of a word considering the larger context cannot be encoded. Frequently, the normalization falls back to averaging over all word embeddings. However, especially for larger texts this results in representations containing less semantics. Both problems are addressed by contextualized word embeddings such as **BERT** [14]. BERT uses masked language models and transformers to predict missing words. Due to its architecture these contextualized word embeddings provide also context embeddings which can be used as representations for larger word sequences. The semantics encoded into both models is based on the empirical usage of the considered words in a training corpus. One special kind of textual embedding model is the **Wikipedia2Vec** [15] approach. Instead of learning only word representations in a similar setting as Word2Vec it jointly learns representations for Wikipedia entities based on predicting neighbours in the Wikipedia link graph. The link graph connects entities that are mentioned in the corresponding articles of other entities and thereby provides additional external knowledge for the representations. Due to the exploited graph structure this is similar to the second modality which is frequently exploited: Knowledge Graph Representations.

3.2. Knowledge Graph Representations

Instead of implicitly retrieving semantics by considering the statistics of a large corpus, Knowledge Graphs provide explicit semantics by defining entities and relations between those entities. Providing explicit knowledge enables a more precise definition of the class representations and thereby improves the understanding. However, only a small number of taxonomies already maintain suitable references to KGs. This makes the entity-linking step described to retrieve additional textual data also necessary for KGs. Similar to the considered corpus for textual representations KG representations depend significantly on which KGs are utilized as external knowledge source. On one side the large nodes of the Linked Open Data cloud, like DBpedia or Wikidata, could be utilized or subject-specific KGs which provided more detailed domain knowledge, but are not available for many domains and tend to contain less knowledge.

The linked entities can be transformed into a low dimensional vector by a wide range of KG embedding models. In the scope of this paper three KG models are considered. **TransE** [16] represents entities of a KG by defining relations of a KG as translations in the embedding space from the head to the tail of triples. More specifically, given a triple `<h,r,t>` the model is trained to build a vector space where $t \approx h \oplus r$ holds. The training is performed by corrupting triples to generate negative samples i.e., the tail (or head) of the triple is substituted by another entity; the model is optimized to distinguish between corrupted and non-corrupted triples. In doing so, given a triple `<h,r,t>` and a corrupted triple `<h,r,t'>` from the generated embeddings space, $h \oplus r$ should be closer to t than t'. In opposition to that **TransR** [17] represents entities and relations in different embedding spaces. The assumption behind this model is that entities and relations in KGs are different objects and, thus, they need two distinct representations. Given a triple `<h,r,t>` the model uses a translation matrix M_r to move h and t from the entity space to the relation space. The score function used by the model is given in equation $f_r(h,t) = \|hM_r + r - tM_r\|_2^2$. **RDF2Vec** [18] is a model which adapts the *Word2Vec* algorithm to graph representations. First, it creates sequences of entities and relations by performing random walks on the graph in order to build sequences that can be used as text sentences. Then, the *skip gram* or the *CBOW* methodologies are applied to build embedding representations of entities and relations.

3.3. Understanding Representation Spaces

In Zero-shot Text Classification the class representations are mapped into a shared class-document space to perform the classification by utilizing distance / similarity metrics. This mapping is learned by aligning the subset of classes with sufficient training data and then applied to unseen class representations. Consequently, the classification is based on the assumption that the relevant semantic relations between classes are encoded in the class embedding space. Therefore, a better understanding of the class representations is provided by evaluating the semantic relatedness between the classes. The straightforward way of analysing the semantic relatedness compares the vector similarity between a pair of embeddings to a human assigned label. This is the same process as for the word semantic similarity task. However, the human labels are highly subjective for class representations as well and require a detailed description on how specific relations should be quantified. These problems can be partially circumvented by comparing the similarity of two pairs. Such an comparison can be defined based on a triple of classes `<Anchor, A, B>` and a label indicating if class A or class B is more similar to the $Anchor$ class. The intrinsic evaluation of an embedding space Θ predicts the label by checking if the constraint $cosine\ similarity(\Theta(Anchor), \Theta(A)) > cosine\ similarity(\Theta(Anchor), \Theta(B))$ is true and can be analyzed by precision, recall and f-measure in a binary classification setting.

Unfortunately, not all class combinations share some kind of relation among them, making it necessary to select only triples where such a relation exists at least for one pair. Additionally, in some cases the semantic similarity could be equal between the pairs. Consequently, the human labels need to include a third value to identify the cases where it is not possible to decide. The evaluation can either filter these cases which reduce the available labels but does not require any additional hyperparameter or introduce a threshold as well as a minimal similarity to predict this class as well and extend the binary classification setting to a multi-class setting.

4. Evaluation

The presented class representations are evaluated based on the proposed intrinsic evaluation on the example of the *arXiv.org* computer science classes. ArXiv manages its many published scholarly articles by categorizing them according to a taxonomy. Thereby, it represents a typical multi-class multi-label classification.

4.1. Gold Standard: ArXiv Classes

The arXiv taxonomy provides for all classes a descriptive name and a brief description. Additionally, the arXiv classes are manually mapped to the most suitable DBpedia entity, which enables retrieving the Wikipedia abstracts as textual description of the given classes. During this step classes like *General Literature* and classes without a suitable DBpedia mapping are filtered. This leaves overall 31 classes. In order to investigate the influence of explicit domain knowledge the classes are also mapped to AI-KG [19], a KG generated based on scholarly articles from the computer science domain. An overview of the amount of data available for the arXiv classes is given in Table 1.

Table 1
Overview of the external data with min, average and max number of tokens for text attributes and number of triples containing the mapped entity for the KGs.

Attribute	Size min; average; max	Example
Name	1; 2.5; 5	Artificial Intelligence
Wikipedia abstract	22; 209.2; 494	Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans and animals. Leading AI textbooks [...]
DBpedia	5; 1772.5; 7766	*dbr:Artificial_intelligence dcterms:subject dbr:Emerging_technologies .* [...]
AI-KG	3; 650.8; 4091	*aikg:artificial_intelligence rdf:type aikg-o:Task .* [...]

The evaluation utilize mostly pre-trained embeddings models. The textual embeddings use the skip-gram Word2Vec model[3] built on the Google News dataset, the pre-trained BERT model[4] built on BookCorpus and the English Wikipedia and finally skip-gram Wikipedia2Vec[5]. The KG entities for DBpedia use the models pre-trained for [20] which are online available[6]. Based on this code also the relevant AI-KG embeddings are trained. Additionally, the multi-class setting uses half of the standard derivation of all class similarities as threshold and the 10th percentile as minimum value.

The gold standard is created by random subsampling all combinations of the 31 arXiv classes and is manually annotated by 11 experts from the computer science domain. The annotators were instructed by a brief task description and were provided the available textual data from arXiv (descriptive name, brief description). Overall the experts labelled $3,000$ triples with 5 votes for each triple. However, the analysis of intra- and inter-annotator agreement indicates only a small reliability of the whole dataset. The intra-annotator agreement is calculated based on 20 triples which were labeled twice by every annotator. The Cohen's κ coefficient for this agreement ranges from 0.14 to 0.9 with an average of 0.49. The inter-annotator agreement is measured by the averaged Cohen's κ coefficient of the 5 votes provided for each triple. It is 0.22. The Krippendorff's α coefficient of 0.21 confirms this low reliability. Evidently, random subsampling includes many controversial triples, where the label depends on minor differences in the mental model of the individual annotators. In order to create a reliable gold standard these controversial triples are filtered out. A triple is considered as controversial if more then one of the votes disagrees with the other votes. After this step the gold standard contains $1,266$ triples with 300 A labels, 354 B labels and 612 triples where no decision was possible.

4.2. Results and Discussion

The results are reported in Table 2. For the binary classification precision, recall and F-measure are presented and for the multi-class classification only the micro F-measure is considered,

[3]https://code.google.com/archive/p/word2vec/
[4]https://huggingface.co/bert-base-cased
[5]https://wikipedia2vec.github.io/wikipedia2vec/pretrained/
[6]https://github.com/nheist/KBE-for-Data-Mining

because the actual class distribution is arbitrary and by definition precision, recall and F-measure are equal in the micro setting. However, both classification settings provide similar results. This indicates that unrelated triples and equally similar pairs are not a special case, which suggests that binary classification with less hyperparameter is sufficient for the intrinsic evaluation.

Overall, the Wikipedia2Vec embeddings align best with the actual semantic similarities, followed by Word2Vec generated from names and BERT using Wikipedia abstracts. All KG embeddings could not reach this performance. Overall, no model is able to reach human-level performance. However, the human results presented are the average of the same annotations used for creating the gold standard and due to this biased. Independent labels would be below these results.

Table 2
Evaluation in terms of Precision, Recall, and F-measure.

		Binary			**Multi-Class**
Model	**Attribute**	Precision	Recall	F-measure	Micro F-measure
Word2Vec	Name	0.668	**0.757**	0.709	0.484
Word2Vec	Wiki abstract	0.682	0.65	0.666	0.478
BERT	Name	0.591	0.593	0.592	0.379
BERT	Wiki abstract	0.658	0.727	0.691	0.495
Wikipedia2Vec	Wiki entity	**0.738**	0.74	**0.739**	**0.563**
TransR	DBpedia	0.548	0.573	0.56	0.415
TransR	AI-KG	0.498	0.55	0.523	0.439
TransE	DBpedia	0.508	0.513	0.511	0.397
TransE	AI-KG	0.501	0.597	0.545	0.42
RDF2Vec	DBpedia	0.496	0.573	0.532	0.356
Human Annotator		0.947	0.853	0.881	0.86

The most apparent insight this analysis provides is the difference between both modalities. KG embeddings in general lack behind textual embeddings. With only small differences between the embedding models and considering that RDF2Vec exploits a similar methodology as Word2Vec this can be explained by the smaller amount of training data. Even large KGs like DBpedia only provide a few thousand triples for each class. A text corpus with significant more mentions of these classes is able to provide a better semantic model.

However, the comparison between Word2Vec and BERT, where BERT represents the larger model trained with more data, shows that the model size is not the only relevant factor. Especially, the BERT model relies on task-specific fine-tuning, which is not possible for unseen classes in Zero-shot Classification. These results are coherent with the extrinsic evaluation of Word2Vec and BERT in [2] where a fine-tuned BERT model performs on a similar level as untrained Word2Vec for unseen classes. Interestingly, the results show the main drawback of a Word2Vec model, too. If larger texts, like the Wikipedia abstracts, are normalized by basic averaging all embeddings get more similar and the performance decreases. On the other hand contextualized models as BERT perform worse without the context.

A less pronounced difference in the results is given between the KG embedding models and the used KG. TransE returns slightly better results on the domain specific AI-KG, but

TransR performs better on a larger dataset with more heterogeneous relations (DBpedia). That illustrates that TransE struggles with heterogeneous relations and therefore benefits from more domain-specific knowledge.

Overall, Wikipedia2Vec as a hybrid model using text and graph embeddings provides the best semantic relatedness. Such models seem to extract the semantics from large textual corpora and use the entity relations to emphasis or add important semantic relations. Thereby, it combines advantages of both modalities.

5. Conclusion

This paper describes an intrinsic evaluation, which provides a better understanding of the class vector space for Zero-shot Text Classification by checking human defined greater-than constraints between classes. Therefore, common embeddings models using textual data and KGs to define classes are generated for the computer science arXiv subset and evaluated with a created gold standard. This investigation shows that textual embeddings are able to extract more implicit knowledge compared to the explicit knowledge provide by KGs and that extending textual embeddings with (Knowledge) Graph-based information is able to capture semantic relatedness better than single modalities.

This line of research can be extended by an empirical study on the correlation between the performance of unseen classes and the semantic relatedness to confirm the theoretical argument for the intrinsic evaluation. Additionally, more embedding models using both modalities should be evaluated with respect to semantic relatedness. Another ongoing effort is the extension of the presented dataset considering more triples and a larger pool of domain experts. This way the intrinsic evaluation of class representations is able to facilitate further research, and thereby improve Zero-shot Text Classification.

References

[1] Y. Song, D. Roth, On dataless hierarchical text classification, in: Proceedings of the 28th AAAI conference on Artificial Intelligence, 2014, p. 1579–1585.

[2] W. Yin, J. Hay, D. Roth, Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, pp. 3914–3923.

[3] J. Zhang, P. Lertvittayakumjorn, Y. Guo, Integrating semantic knowledge to tackle zero-shot text classification, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 1031–1040.

[4] F. Hoppe, D. Dessì, H. Sack, Deep learning meets knowledge graphs for scholarly data classification, in: Companion Proceedings of the Web Conference 2021, 2021, pp. 417–421.

[5] M. W. Chang, L. Ratinov, D. Roth, V. Srikumar, Importance of semantic representation: Dataless classification, 27th AAAI conference on Artificial Intelligence (2008).

[6] F. Hoppe, T. Tietz, D. Dessì, N. Meyer, M. Sprau, M. Alam, H. Sack, The challenges of German archival document categorization on insufficient labeled data, in: Proceedings of the Third Workshop on Humanities in the Semantic Web, co-located with 15th Extended Semantic Web Conference, 2020, pp. 15–20.
[7] S. Bujwid, J. Sullivan, Large-scale zero-shot image classification from rich and diverse textual descriptions, in: Proceedings of the Third Workshop on Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN), 2021, pp. 38–52.
[8] A. Bakarov, A survey of word embeddings evaluation methods, arXiv (2018).
[9] N. Jain, J.-C. Kalo, W.-T. Balke, R. Krestel, Do embeddings actually capture Knowledge Graph semantics?, in: 18th Extended Semantic Web Conference, 2021, pp. 143–159.
[10] B. Chiu, A. Korhonen, S. Pyysalo, Intrinsic evaluation of word vectors fails to predict extrinsic performance, in: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, 2016, pp. 1–6.
[11] F. F. Liza, M. Grześ, An improved crowdsourcing based evaluation technique for word embedding methods, in: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, 2016, pp. 55–61.
[12] F. Hill, R. Reichart, A. Korhonen, Simlex-999: Evaluating semantic models with (genuine) similarity estimation, Computational Linguistics (2015).
[13] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv (2013).
[14] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv (2018).
[15] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, Y. Matsumoto, Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 23–30.
[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems (2013).
[17] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for Knowledge Graph completion, in: 29th AAAI conference on Artificial Intelligence, 2015, pp. 2181–2187.
[18] P. Ristoski, H. Paulheim, Rdf2Vec: RDF graph embeddings for data mining, in: International Semantic Web Conference, 2016, pp. 498–514.
[19] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, E. Motta, H. Sack, AI-KG: an automatically generated knowledge graph of artificial intelligence, in: International Semantic Web Conference, 2020, pp. 127–143.
[20] J. Portisch, N. Heist, H. Paulheim, Knowledge Graph embedding for data mining vs. Knowledge Graph embedding for link prediction–two sides of the same coin?, Semantic Web Journal (2021).