

IRT2: Inductive Linking and Ranking in Knowledge Graphs of Varying Scale

Felix Hamann^{1,*}, Adrian Ulges¹ and Maurice Falk¹

¹RheinMain University of Applied Sciences, Wiesbaden, Germany

Abstract

We address the challenge of building domain-specific knowledge models for industrial use cases, where labelled data and taxonomic information is initially scarce. Our focus is on inductive link prediction models as a basis for practical tools that support knowledge engineers with exploring text collections and discovering and linking new (so-called open-world) entities to the knowledge graph. We argue that – though neural approaches to text mining have yielded impressive results in the past years – current benchmarks do not reflect the typical challenges encountered in the industrial wild properly. Therefore, our first contribution is an open benchmark coined IRT2 (inductive reasoning with text) that (1) covers knowledge graphs of varying sizes (including very small ones), (2) comes with incidental, low-quality text mentions, and (3) includes not only triple completion but also ranking, which is relevant for supporting experts with discovery tasks.

We investigate two neural models for inductive link prediction, one based on end-to-end learning and one that learns from the knowledge graph and text data in separate steps. These models compete with a strong bag-of-words baseline. The results show a significant advance in performance for the neural approaches as soon as the available graph data decreases for linking. For ranking, the results are promising, and the neural approaches outperform the sparse retriever by a wide margin.

Keywords

Text Mining, Knowledge Graphs, NLP,

1. Introduction

Knowledge graphs (KG) are known to be powerful tools in various applications such as web search or personal assistants. They are key to addressing complex information needs that require reasoning and cannot be answered with simple text matching. Usually, knowledge graphs are assumed to be available at large scale [1], examples include the Linked Open Data Cloud¹ or KGs in biomedicine [2]. However, knowledge graphs may also be interesting in other industries where no large-scale knowledge exists yet, be it in the technical service, where KGs can help reason about the root causes of machine failures, in insurance, where they can model the details of a contract, or in finance rating, where they capture businesses' supply chains.


Proceedings of the Workshop on Text Mining and Generation (TMG) co-located with the 45rd German Conference on Artificial Intelligence (KI 2022), September 19, 2022, Trier (Virtual), Germany


*Corresponding author.

✉ felix.hamann@hs-rm.de (F. Hamann); adrian.ulges@hs-rm.de (A. Ulges); maurice.falk@hs-rm.de (M. Falk)

🌐 <https://www.cs.hs-rm.de/~hamann> (F. Hamann); <https://www.cs.hs-rm.de/~ulges> (A. Ulges)

🆔 0000-0002-7539-0432 (F. Hamann)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://lod-cloud.net>

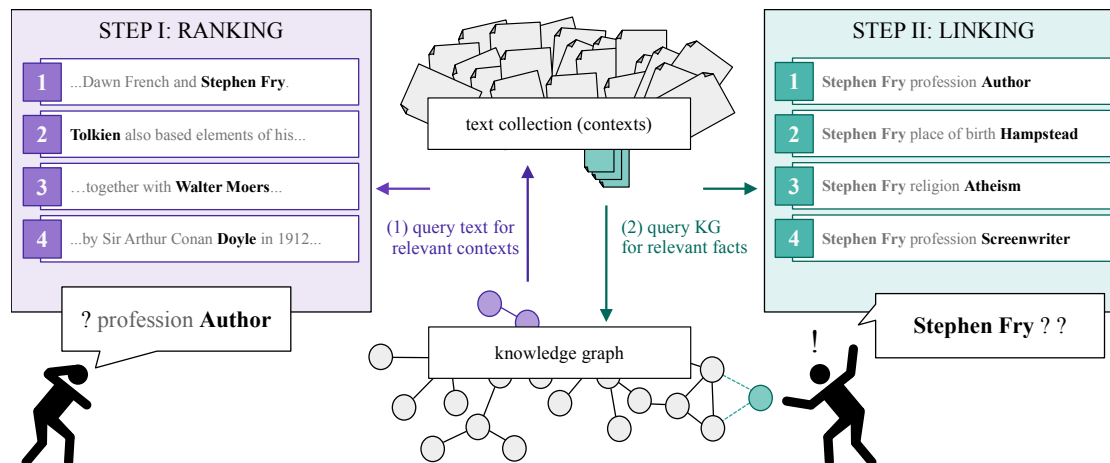


Figure 1: Our two-step approach towards interactive knowledge acquisition: First, the user discovers sentences containing relevant entities (Discovery/Ranking, left). Second, given a mention of interest, triples for this mention are suggested as if it was an entity (Linking, right).

Explicit knowledge is scarce in these domains. However, what is often commonly available in abundance is text data, such as service tickets, insurance claims, or business reports.

The focus of this paper is to support experts with building knowledge graphs by inspecting text. To do so, we assume that the expert adds triples to the graph in an iterative process, in which he/she locates and identifies new entities in the text collection and links them to the graph. We address two key steps of this process, as illustrated in Figure 1:

1. **Discovery/Ranking:** The expert explores the text collection based on a partial fact of the KG (i.e. a relation-vertex tuple). He/she specifies an information need (such as “find me actors”, or more specifically “find sentences likely to contain mentions of entities which have the relation `profession` with the known entity `actor`”). The system returns a ranked list of potentially interesting sentences, giving rise to a ranking task.
2. **Linking:** The expert studies the sentences. Once he/she has discovered an interesting entity x (such as **Frances McDormand**), we would like to link x to the graph by adding triples, not only via the `profession` relation but also via other relations (such as the birthplace, languages she speaks, or movies she appeared in). To do so, the system collects textual evidence on the entity in form of other mentions in the text collection. From those, it infers triples that link x to the knowledge graph.

We address both tasks using embedding-based *semi-inductive* [3] *open-world* [4] link prediction, which is targeted at predicting the likelihood of triples (h, r, t) . In contrast to standard link prediction, semi-inductive link prediction addresses *new* entities to be linked to the known graph and the open-world scenario connotes that the new entities are solely described via free text.

Since domain-specific data is often confidential, link prediction research employs open data from graphs such as Freebase [5] or Wikidata [6]. We argue that the insight these benchmarks offer for industrial knowledge acquisition is limited, and contribute a new benchmark called

Inductive Reasoning with Text V2 (IRT2) with the following benefits²: (1) To assess models at different stages of graph construction, we benchmark on graphs of varying size. (2) While text contexts in other datasets consist of concise descriptions of entities, text in practice contains rather incidental mentions of entities. We sample our text data accordingly. (3) To our knowledge, our study is the first to not only address the linking task but also the ranking task.

We evaluate three inductive link models on IRT2: One baseline based on keyword matching, and two neural models that combine a transformer text encoder with a link predictor. We show that the end-to-end approach and separate training both work well, while the latter generally performs a little better (even for small graphs). Also, only through evaluation on smaller graphs, the baseline model is outperformed when linking, which favours the neural models in scenarios where structured data becomes sparse.

2. Related Work

Link Prediction Models: Closed-world link prediction (or also *knowledge graph completion*, *KGC*) has attracted interest in the research community over the past years. Earlier approaches such as RESCAL [7], TransE [8], and DistMult [9] laid the foundations for many following completion models such as ComplEx [10], ConvE [11], RotatE [12], or KBGAT [13]. We are, however, interested in open-world scenarios, where, given a textual description of an entity, geometric reasoning is applied through the alignment of dense graph- and text-representations. This combines KGC—which aims at the prediction of missing links between existing entities—with language modelling [14, 15, 16] which produces dense vector representations for natural language text. Specifically, in this work, a semi-inductive setting [3] is studied, where out-of-kg entities are to be linked to an existing graph using their textual description. One of the first approaches to combining language- and graph-reasoning models is NTN [17], where entity description embeddings are trained to be similar if their entities are connected in the KG. Later approaches build on this idea: DKRL [18], ConMask [4], MIA [19], and KEPLER [20] introduce models that jointly learn the connection between entity and graph representations. A different approach decouples the KGC scorer and text encoder such that first the scorer is trained independently and in a separate step a projection from text-based entity descriptions to their associated graph-based embeddings is learned [21, 22, 23]. We study and compare both approaches with the models described in Section 4. Highly related to our work is the recently published approach by Daza et al [24]. Here, a BERT model (BLP) is trained to directly score KGC triples for plausibility using the textual representations instead of training a separate entity embedding. However, we cannot rely on high quality text and thus a separate entity embedding tuned with many different observations for input text is more suitable.

KGC Benchmarks: Closed-world KGC models are usually evaluated on subsets of publicly available knowledge graphs such as Freebase [5], DBPedia [25] or Wikidata [6]. Famous benchmarks include FB15K [8], succeeded by FB15k237 [26] due to test-leakage, WN18, succeeded by WN18RR [11], or CoDEX [27], among others. Open-world KGC combines KGs with textual descriptions of its entities. Approaches include the FB20K benchmark as part of DKRL [18], DBPedia50k and DBPedia500k as part of ConMask [4], and FB15k237-OWE as part

²The data is publicly available under <https://github.com/lavis-nlp/irt2>

of OWE [21]. More recent work introduces Wikidata5M [20] (a large Wikidata subset) and the FB15k237/WN18RR adaptations of [24]. Since data in industrial use cases is not publicly available, we also sample our benchmark data from open knowledge graphs (CoDEX) and text sources (Wikipedia). However, all the above benchmarks tackle the open-world scenario with a single concise description of graph entities. We are, however, interested in linking entities associated with many, noisy text contexts. To our knowledge, the only benchmark to attempt this is IRT [28]. Here, a set of 30 text contexts with incidental mentions is associated with each entity in the knowledge graph. We extend this benchmark by offering more text, multiple graph sizes, a greater variety of mentions, and evaluation protocols for both ranking and linking.

3. Problem Description

We define a knowledge graph (KG) as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T}, \mathcal{M}, \mathcal{C})$ where \mathcal{V} is the set of vertices v (e.g. $v = \text{Fargo}$) and \mathcal{R} a set of relation types (e.g. $r = \text{genre}$). Triples $(h, r, t) \in \mathcal{T} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ connect the vertices, e.g. $(\text{Fargo}, \text{genre}, \text{Thriller Film})$. Additionally, the set \mathcal{M} contains textual mentions, each associated with a vertex via a function $M : \mathcal{V} \mapsto \mathcal{P}(\mathcal{M})$. For example, the entity $v = \text{Thriller Film}$ has the mentions $M(v) = \{ \text{“crime thriller”}, \dots, \text{“gangster film”} \}$. Note that mentions can be ambiguous (e.g. “the film” as mention of **Fargo**). Furthermore, each of the mentions m is assumed to occur in textual context sentences c from a corpus \mathcal{C} . In general, these contexts do not express triples but solely contain incidental mentions of entities. They are accessible through a function $C : \mathcal{V} \times \mathcal{M} \mapsto \mathcal{P}(\mathcal{C})$. For example, with $v = \text{Crime Film}$ and $m = \text{“gangster film”}$: $C(v, m) = \{ \text{“Corman called it the most accurate, authentic gangster film”}, \dots \}$. Note that – via M and C – we can access an entity’s mentions and contexts, and vice versa.

Open/Closed-World: While the text corpus \mathcal{C} contains known (or "closed-world") entities, a second corpus \mathcal{Q} contains undiscovered open-world mentions \mathcal{M}^o . Our goal is to discover these and link them to the graph. We define a closed-world graph \mathcal{G}^c (containing subsets of the respective sets of \mathcal{G} and the text corpus \mathcal{C}) and an open-world graph $\mathcal{G}^o = (\mathcal{V}^o, \mathcal{R}, \mathcal{T}^o, \mathcal{M}^o, \mathcal{Q})$ with $\mathcal{V}^o \subseteq \mathcal{V}$, $\mathcal{T}^o \subseteq \mathcal{T}$, $\mathcal{M}^o = \mathcal{M} \setminus \mathcal{M}^c$: Namely a graph with a set of undiscovered mentions not known in the closed-world graph \mathcal{G}^c . This gives rise to the two tasks we address:

Ranking Task: The first task is to retrieve text contexts from \mathcal{Q} which contain mentions of interest, i.e. unknown mentions of (possibly unseen) entities which should complete a given entity-relation pair. For example, for relation $r = \text{genre}$ and tail $t = \text{crime film}$, we search for text contexts where *any* crime films are mentioned.

Linking Task: Now, given an open-world mention was just discovered and marked, all text contexts of \mathcal{Q} are bundled by mention. For each such bundle of text, links $r \in \mathcal{R}$ to the graph vertices V^c must be found. For example: *“What genre is associated with all text contexts that contain the mention fargo?”*.

4. Models

We tackle the above tasks with three models: (1) **JOINT**, where a text encoder and a knowledge graph completion (KGC) scorer are trained jointly, (2) **OWE**, where an encoder learns to project

text representations into a pre-trained graph embedding space, and (3) **BOW**, a model that employs bag-of-words representations for text similarity. All three models are evaluated on both the ranking and linking tasks. Note that, although our notation is (for brevity) generally focused on finding tails, we do also always include head-prediction for given relation-tail pairs.

4.1. Neural Models

Both neural models combine a text encoder ϕ and a KGC module ψ (see Figure 2). As a text encoder, we use a pre-trained BERT [16], a popular state of the art transformer [29] encoder. Given a text context c , we run it through the encoder and select the [CLS]-token embedding as the context representation $\phi(c)_{\text{CLS}} \in \mathbb{R}^{d'}$ (following the observations of [28]). This representation is mapped using a learned affine projection (W, \mathbf{b}) into a complex-valued space, obtaining a representation \mathbf{c} . The representation is then used to estimate a tripel’s plausibility. We use the ComplEx scorer [10], which, given complex-valued context, relation and tail embeddings $\mathbf{c}, \mathbf{r}, \mathbf{v} \in \mathbb{C}^d$, calculates the plausibility score as $\psi(c, r, v) = \mathbf{Re}(\mathbf{c} \odot \mathbf{r} \odot \bar{\mathbf{v}})$. Note that—since the projection maps from real-valued to complex space—we choose the matrix and bias vector dimensions accordingly ($W \in \mathbb{R}^{d' \times 2d}, \mathbf{b} \in \mathbb{R}^{2d}$). Overall, our neural models can be written as:

$$s(c, r, v) = \psi(\underbrace{W \cdot \phi(c)_{\text{CLS}} + \mathbf{b}}_{\mathbf{c}}, \mathbf{r}, \mathbf{v}) \quad (1)$$

Multi-Context Extension As discussed above, instead of a single context c , a set Σ of *multiple* contexts may be available describing the same entity. We would like our entity representations to take the whole set Σ into account. To do so, we use a simple early fusion that averages the text representations before projecting them:

$$\mathbf{c} = W \cdot \left(\frac{1}{|\Sigma|} \sum_{c \in \Sigma} \phi(c)_{\text{CLS}} \right) + \mathbf{b} \quad (2)$$

We then use the resulting representation \mathbf{c} as a drop-in replacement in Equation (1). Concerning training, we use two variants of this setup: In the first variant (referred to as **single**), training happens on individual contexts as in Equation (1), but we apply Equation (2) in inference. In the second variant (**multi**), we apply Equation (2) both in training and inference.

Training

As illustrated in Figure 2, we study two different training strategies to fit the model parameters (text encoder ϕ , projection W, \mathbf{b} , and knowledge graph embeddings \mathbf{r}, \mathbf{v}). Our first approach uses a joint (end-to-end) training (**JOINT**): We iteratively sample a random entity v from the knowledge graph. For v , we draw a random triple (v, r, v') and context c . The model’s scores $s(c, r, v')$ are mapped to a probability distribution using a softmax, and we apply a cross-entropy loss:

$$\mathcal{L}_{\text{JOINT}} = - \sum_{v' \in \mathcal{V}^c} \left(\log \frac{\exp(s(c, r, v'))}{\sum_{u' \in \mathcal{V}^c} \exp(s(c, r, u'))} \right) \cdot \mathbf{1}_{(v, r, v') \in \mathcal{T}} \quad (3)$$

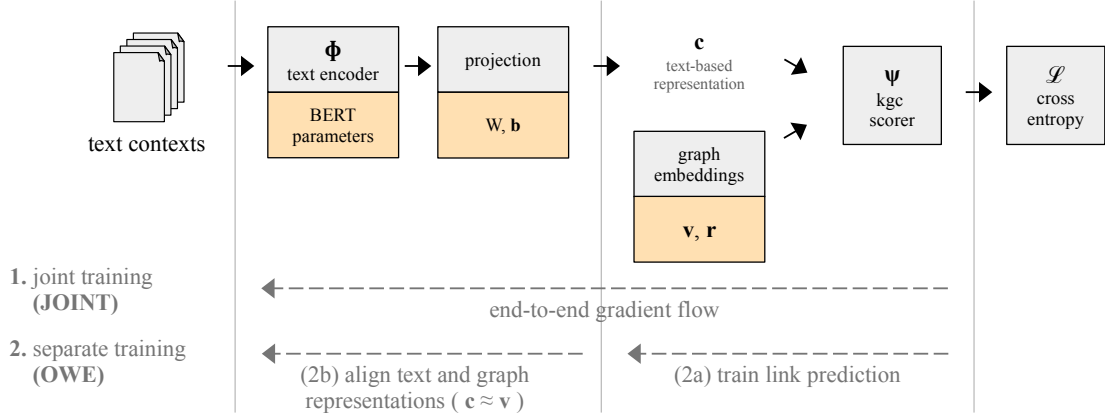


Figure 2: Data flow in our model (top) and the two training strategies (bottom): Strategy 1 (JOINT) applies an end-to-end training of both link predictor and text encoder. Strategy 2 (OWE) first trains the link predictor (2a), and then aligns the text-based entity representations \mathbf{c} with the graph-based ones \mathbf{v} (2b). Learnable parameters are highlighted in orange.

Note that – though the above notation samples triples (v, r, v') to predict tails v' – we also sample triples (v', r, v) and predict the heads v' accordingly.

Our second approach follows Shah et al’s OWE model [21] and applies a **separate training**. Here, training happens in two steps: First, the link prediction model ψ is trained on the closed-world graph, obtaining a *graph-based embedding space*. Second, the text encoder ϕ and projection (W, \mathbf{b}) are trained to align entities’ text-based representations with the (now fixed) graph-based ones. To do so, we reduce the geometric distance between context representations $\mathbf{c} = W \cdot \phi(c)_{CLS} + \mathbf{b}$ and their associated graph-based representations, using a mean squared error loss:

$$\mathcal{L}_{\text{OWE}} = \frac{1}{2d} \cdot \sum_{0 < i \leq d} (\text{Re}(\mathbf{c}_i) - \text{Re}(\mathbf{v}_i))^2 + (\text{Im}(\mathbf{c}_i) - \text{Im}(\mathbf{v}_i))^2 \quad (4)$$

Inference: This section outlines how the neural models are used in our two tasks. First, in **ranking**, a closed-world entity v and a relation of interest r are given, and the task is to retrieve a ranked list of text contexts from the query corpus \mathcal{Q} that potentially contain mentions of relevant open-world entities to be identified by the expert. To do so, we compute the single-context scores $s(c, r, v)$ for all contexts $c \in \mathcal{Q}$, $r \in \mathcal{R}$ and $v \in \mathcal{V}^c$. The scores are normalized per relation using the softmax function as to better compare the independently scored contexts c (the raw ComplEx scores are unbounded). The contexts in the query corpus are then ranked by this normalized score.

Second, in **linking**, the expert is interested in a certain mention of interest $m \in \mathcal{M}^o$ representing an open-world entity. His/her goal is to link said entity to the graph. To do so, he/she selects a relation $r \in \mathcal{R}$ and collects a set of contexts Σ containing m to describe the open-world entity. Using the contexts, all entity candidates $v \in \mathcal{V}^c$ are ranked by their scores $s(\Sigma, r, v)$. Note that in principle we can repeat this procedure for each relation and also for predicting heads instead of tails.

4.2. BOW: Bag-of-Words Retrieval

To evaluate the tasks at hand with a common industrial approach, we compare the above neural models with a bag-of-words approach. Given one or several mentions, this approach concatenates text contexts containing the mentions into *documents* and then conducts a similarity matching between documents using the well-known BM25[30] keyword matching implemented by the Elasticsearch search engine³.

Ranking: Given an entity relation pair such as $v = \text{Thriller Film}$, $r = \text{genre}$ (“Find text contexts which mention thriller films”), the baseline’s strategy is to find open-world entities that are similar to known closed-world entities v' for which (v', r, v) holds (i.e., known thriller films). We sample a set of random *representatives* v' and some of their associated text contexts. These contexts are concatenated into a document which is then used as a query against an index containing all open-world contexts \mathcal{Q} .

Linking: Given a mention $m \in \mathcal{M}^o$ and a relation r , we sample contexts containing m into a document $D(m)$, which is used to query against an index containing documents $D(v)$ for all closed-world vertices $v \in \mathcal{V}^c$. The top-n predicted results allow us to compile a set of reference vertices which we use as blueprints for prediction. For example: When searching for $r = \text{profession of text contexts containing “s. a. corey”}$ (“Predict the profession of the text contexts containing s. a. corey”), the result list may contain other texts describing authors. The final prediction of vertices is compiled from the retrieved vertices targets of relation r and scored by the inverse of the position of the document in the result list.

5. Benchmark Construction

Our benchmark picks up the work of [28] and extends it: (1) We study the behaviour of models with varying knowledge graph size, and particularly for small graphs, (2) we test models for both the ranking and linking task. We offer four variants of the benchmark: **tiny**, **small**, **medium**, and **large**. The variants offer different ratios of structural information (i.e. graph triples) and associated text. The datasets are based on CoDEx-M [27] which is sampled from Wikidata [6]. The following steps are executed:

1. Identifying Concept Vertices: We assume that in a real-world scenario *world-knowledge* is more likely to be already modelled in the given KG. Information to be discovered is usually much more volatile. Consider for example a graph for a machine manufacturer. We know that things can catch fire (the world knowledge) but we aim to find the specific parts that actually do. A heuristic to identify such world knowledge is based on the assumption that such entities are characterised by relations which exhibit a strong disproportion between their heads and tails. The ratio between a relation r ’s domain size (its number of heads) $\text{dom}(r) := |\{h \mid \exists t : (h, r, t) \in \mathcal{T}\}|$ and the range size (its number of tails) $\text{rg}(r) := |\{t \mid \exists h : (h, r, t) \in \mathcal{T}\}|$ is $\min(\text{dom}(r), \text{rg}(r)) / \max(\text{dom}(r), \text{rg}(r))$. Per size variant of the dataset, we select a subset of the relations offered by CoDEx, order by ratio and select a subset of those to determine *concept entities* that will remain in the closed-world split. The appendix in ?? enumerates all relations and corresponding selections in greater detail.

³<https://www.elastic.co>

2. Sampling Mentions and Text Contexts: As our graph is based on Wikidata, we can exploit its links to Wikipedia. For each vertex v , we identify the associated Wikipedia page p_v and all pages which link back to it $\mathcal{N}(p_v)$. Using the link text of the hyperlinks, we build the set of mentions which are associated with the vertex $M(v)$. Lastly, to build the contexts $C(v, m)$, all occurrences of the mentions in $\mathcal{N}(p_v)$ are identified and the surrounding sentence kept as context. This is a heuristic to obtain incidental text contexts: The entity is mentioned but usually not the *subject* of the sentence. As example for $v = \text{Scotland}$: “*Aberdeen is a city in northeast Scotland*”.

3. Open/Closed-World Split: The open/closed-world split, in this case, is not done on vertex but mention level. We separate the mention set \mathcal{M} into a partition \mathcal{M}^c for closed-world- and \mathcal{M}^o for open-world mentions respectively. First, all mentions associated with concept entities are set aside for the closed-world part. Then, for each remaining vertex v , its associated mentions are distributed randomly between open- and closed-world. We set an additional pruning parameter which limits the maximum amount of mentions allowed for a closed-world vertex. With the information about closed-world mentions, we identify all triples whose vertices are associated with them and set them aside as the closed-world triple set $\mathcal{T}^c \subseteq \mathcal{T}$. The so-called *test triples* ($m \in \mathcal{M}^o, r \in \mathcal{R}, v \in \mathcal{V}^c$) (explained below the table) are derived directly from the triple set \mathcal{T} by identifying in which triples the mention’s vertex occurs. We set aside a subset of the open-world split for validation. The above steps are executed for the four different dataset variants. These have different parameters set for concept relations, total relations to keep and closed-world mention pruning (see the appendix for more details).

	Tiny	Small	Medium	Large
Relations $ \mathcal{R} $	5	12	45	45
Entities $ \mathcal{E} $	1,174	2,887	3,592	9,952
Training Triples $ \mathcal{T}^c $	2,928	7,527	26,335	102,289
Training Text $ \mathcal{C} $	9,100,422	15,117,184	17,398,943	18,654,485
Test Text $ \mathcal{Q} $	6,058,557	2,390,017	1,905,151	864,598
Ranking Queries	695	1,225	4,972	7,336
Linking Queries	47,857	52,654	97,921	48,801
Test Triples	102,866	102,616	174,312	90,780

The last four rows of the table describe the scale of the challenges. A **ranking query** is a tuple $(v \in \mathcal{V}^c, r \in \mathcal{R})$ (e.g. (**writer**, profession) – “*Find texts with writers in them*”) and an associated ground truth set of open-world mentions $\{m_1, \dots\} \in \mathcal{M}^o$ to be discovered (e.g. { *Tolkiens*, *the author*, *Corey*, ...}). A **linking query** is a tuple $(m \in \mathcal{M}^o, r \in \mathcal{R})$ (e.g. (*Tolkiens*, profession) – “*What professions does the entity have, given texts in which “Tolkiens” occurs?*”) and a set of associated closed-world vertices $v \in \mathcal{V}^c$ (e.g. { **author**, **linguist**, ...}). One can see that both ranking and linking are different views of the same data: unique $(m \in \mathcal{M}^o, r \in \mathcal{R}, v \in \mathcal{V}^c)$ triples. These triples are presented as *test triples* in the table above and allow to gauge the ratio of the queries and their associated ground truth. For example, in the most extreme case, ranking on the tiny variant, has $102866/695 \approx 148$ mentions to discover per query on average.

6. Experiments

We run a series of experiments to build a solid understanding of the challenge’s difficulty. We employ all three formerly described models (JOINT, OWE, BOW) for both tasks (ranking, linking) on all four splits (tiny, small, medium, and large). We measure hits@k and the mean reciprocal rank (MRR) for each test triple (i.e. micro-averaged). Following common practice, we apply target filtering [21] to the scored predictions (mentions for ranking and vertices for ranking). Target filtering is a method where, for a given ranking, only a single true positive of interest is inspected, while all other true positives are removed from the list. This helps to not artificially worsen the result for rankings with many true positives. We offer the datasets, dataset creation code, and evaluation protocol online⁴. The model implementations and all presented trained models are also supplied separately⁵. We evaluate our models on a subset of the test text contexts \mathcal{Q} (400k sentences drawn randomly for ranking; 100 sentences per mention for linking).

6.1. Hyperparameters

We determine the final hyperparameters using a combination of random-, and grid searches over the parameter space. A single closed-world ComplEx model is trained per dataset split and commonly used to train the **OWE** model using Adagrad [31]. The hyperparameters studied comprise learning rate, L2 regularization weight, and embedding space dimensions. We employ PyKEEN[32] for training and build our own open-world extension. We implement and train the **JOINT** and **OWE** models using PyTorch [33] and PyTorch Lightning[34]. Models are optimized using Adam[35]. For each model architecture and dataset split, we run a random parameter sweep to fix a subset of parameters and subject the remaining to a grid search. We study the effect of regularization of the entity and relation embeddings, weight-decay and learning rate of the optimizer, how many layers of the encoder are frozen during training, how many text contexts are sampled per vertex per training, and whether the mention is masked during training. Additionally, we study how many contexts to provide per optimizer step for multi-context models. Generally, all models share a similar set of values for learning-rate, weight-decay and regularization weight. The other parameters are dependent on the split. We found to freeze parts of the encoder works well against overfitting. Contrary to the observations of [28], masking did not significantly increase model performance. The final hyperparameters are provided in the appendix (??) and with the models online.

6.2. Linking

First, we evaluate the linking task as the designated challenge for the presented models. Table 1 details our findings. Generally, a few trends can be observed: (1) The neural models significantly outperform the baseline approach on the three smaller variants but yield inferior results on the large dataset. This supports our claim that a benchmark should offer insights into varying degrees of structural data scarcity. Even with an abundance of data (usually favouring neural

⁴<https://github.com/lavis-nlp/irt2>

⁵<https://github.com/lavis-nlp/irt2m>

approaches), the BOW baseline can beat the neural model’s performance on large but falls short for the smaller variants. (2) Overall, the OWE approach outperforms JOINT albeit not by a great margin for tiny, small, and large. The biggest performance gap can be observed on medium, where the OWE model profits the most from the decoupled training. We argue that this hints towards the need for better sampling/overfitting strategies during training for JOINT as it seems to struggle with the lower variety of mentions seen during training. (3) Although the multi-context models generally perform better than the single-context counterparts, the difference in performance is not as pronounced as reported by [28]. Contrary to their findings we also did not observe much performance improvement through masking the mention. These observations indicate that both models struggle to incorporate much of the textual clues and instead rely heavily on the structural information present in the vertex- and relation embeddings. Overall, the metrics show that the models can link an entity described by text contexts to a given graph with relatively high precision.

		HITS@10				MRR			
		Tiny	Small	Med.	Large	Tiny	Small	Med.	Large
BOW		53.82	55.18	46.43	71.38	33.63	34.62	29.81	50.61
JOINT	single	72.06	70.20	47.14	65.75	50.61	45.95	33.72	48.29
JOINT	multi	73.56	74.27	53.77	65.12	51.28	52.39	37.50	45.26
OWE	single	74.09	74.33	61.98	64.27	50.25	50.57	40.60	42.69
OWE	multi	75.39	71.49	64.41	66.36	53.06	47.17	43.25	45.51

Table 1

Model performance on the **linking** task. The bag-of-words model outperforms the neural approaches on the large dataset. The neural models both outperform the baseline on the smaller variants by a large margin. Among the neural models, OWE generally yields better performance on all splits.

6.3. Ranking

Secondly, for ranking, we study whether the models which were originally trained to do link-prediction can be employed for discovery, too. We report the hits@100 performance in Table 2. (1) The results show that the scoring mechanism of the neural models outperforms the BM25 search results. At first glance, this is a surprising insight, but can be explained by the nature of text samples. As the mention of interest is usually not the subject of the text context, the query does not directly describe its properties. This leads the ranker astray. For example: searching for other comedians (i.e. ?, profession, **comedian**) the model uses text samples sampled from the Wikipedia page about New York among others. This leads to other text samples to be retrieved which also talk about things in the context of New York but seldom other comedians. This *contextualization* of queries (by queried relation) is better considered with the neural models. (2) Overall ranking quality is not high enough for practical application.

We suspect two main factors which hinder effective ranking: First, the scores are calculated independently from each other. This makes an intra-sample comparison of scores difficult. Secondly, the ranking task requires scores first and foremost produced by head prediction. These scores usually are of lower quality because the models are much better at predicting a

		HITS@100			
		Tiny	Small	Med.	Large
BOW		2.86	4.29	6.42	14.83
JOINT	single	7.91	6.78	6.37	19.47
JOINT	multi	13.28	16.17	14.38	30.68
OWE	single	6.30	8.19	6.88	10.81
OWE	multi	9.98	13.00	6.36	31.40

Table 2

Results for the **ranking** task. The neural models all outperform the baseline and, contrary to the linking task, the multi-context JOINT model is generally the best performing approach.

few biased tails (e.g. professions) than hundreds or thousands of volatile heads (e.g. authors). This is worsened by the formerly described observation that text contexts have a comparatively small influence on the score. To obtain better performance, new approaches must be devised which incorporate the graph context on the one hand and neural semantic retrieval [36, 37] on the other.

7. Conclusion

We present IRT2, a benchmark to study a knowledge acquisition pipeline to extend a knowledge graph (KG) from noisy text. We offer knowledge graphs of different sizes and text associated with its vertices. The benchmark comprises two tasks, *ranking* and *linking*, where first, entities need to be discovered from unlabelled text, and in the second stage, linked to the KG. Alongside, we study three approaches for the tasks at hand: two neural open-world triple scorers and a bag-of-words model. We found the models to perform well for the linking task but less so for ranking. For linking, the BOW baseline outperforms the neural models on the large variant. For the remaining, smaller variants, the OWE approach outperforms all other approaches even if graph data is scarce. For ranking, the JOINT model produces the best results. Although the models show a promising direction to be used for ranking, they do not perform at the level for practical application yet. We recommend studying how semantic ranking techniques can profit from the given links between graph- and text data. We invite the community to use the benchmark to (1) devise ways to better incorporate the noisy but abundant text data to make better predictions, (2) find ways to contextualize semantic retrievers with the specific graph information, and (3) derive insights for application in an industrial context with similar constraints and conditions. The dataset and evaluation scripts can be found here: <https://github.com/lavis-nlp/irt2>.

Acknowledgments

This work was supported by the BMBF program FH-Kooperativ, project SCENT (13FH003KX0)

8. Appendices

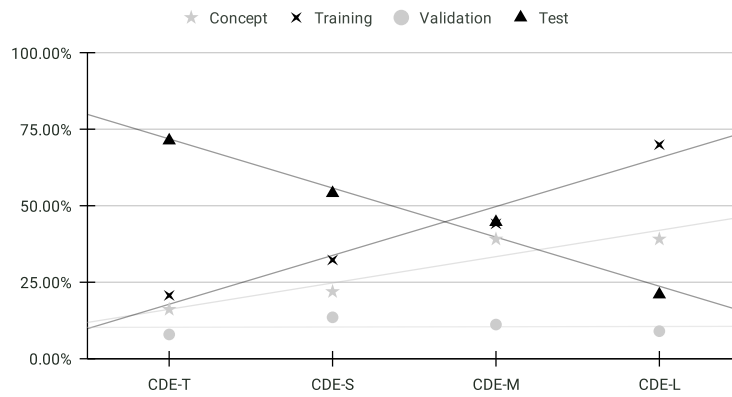
This appendix details configuration options used for benchmark construction and model training in detail to allow for the reproduction of the reported results.

8.1. Benchmark

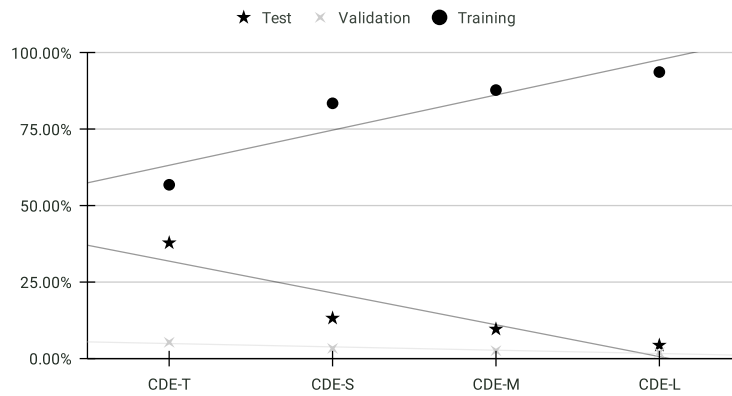
Split	Tiny	Small	Medium	Large
Concept Relations	4	8	27	27
Total Relations	5	12	45	45
Closed World Threshold	400	800	800	-
Target Mention Split	70%	70%	70%	70%
Target Validation Split	10%	20%	20%	80%
Mention Threshold	5	5	5	5
Concept Vertices	471	962	2,200	2,200
Training Vertices	1,174	2,887	3,592	9,952
Training Mentions	4,945	9,231	14,417	22,866
Training Triples	2,928	7,527	26,335	102,289
Training Text Contexts	9,100,422	15,117,184	17,398,943	18,654,485
Validation Vertices	1,741	3,375	3,279	2,644
Validation Mentions	1,894	3,870	3,649	2,940
Validation Vertex Triples	13,639	24,764	43,532	35,209
Validation Task Triples	11,588	25,722	43,716	38,582
Validation Text Contexts	850,351	597,050	505,979	383,351
Test Vertices	10,909	10,527	10,221	5,607
Test Mentions	17,055	15,481	14,600	6,860
Test Vertex Triples	70,094	71,360	124,050	71,951
Test Task Triples	102,866	102,616	174,312	90,780
Test Text Contexts	6,058,577	2,390,017	11,705,151	864,598

The final benchmark proportions rely heavily on the selected relations and pruning options for construction. The following table enumerates both the configuration which was used for sampling and the resulting key figures for the training, validation and test splits. Relations are picked manually per split, are ordered by their ratio and a subset is selected as concept relations. The *Closed World Threshold* determines how many mentions per relation are retained at most in the closed world split. The *Mention Threshold* says to keep only mentions with at least that many associated text contexts. *Validation-* and *Test Vertex Triples* are the triple sets (vertex, relation, vertex) from which the final *Task Triples* are derived (mention, relation, vertex). They are pruned by filtering out *true* open-world vertices (the necessary zero-shot entity linking is out of scope for the presented tasks)—which explains the decrease that can be observed for the tiny variant.

Mention Distribution

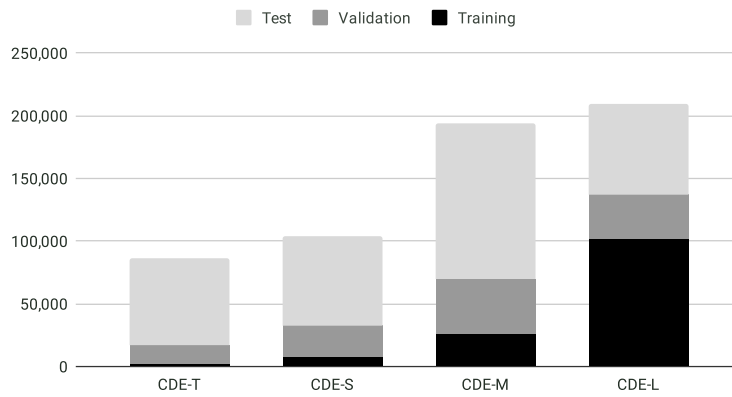


Text Context Distribution

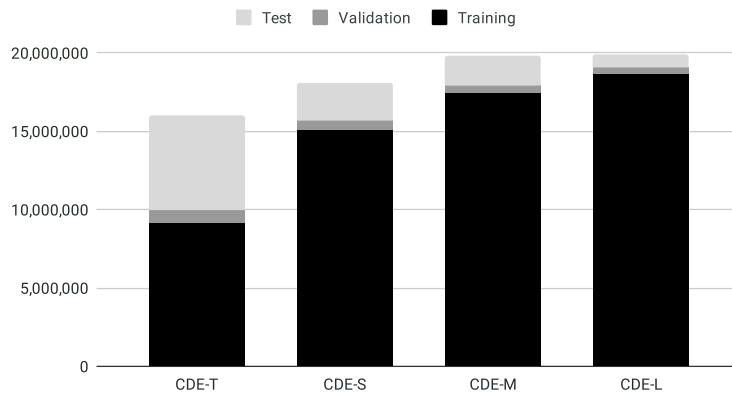


The above figures detail the shift in proportions between the dataset variants. The test mention counts decrease proportionally to the increase in training. Concept mention count increases and is equal for M and L. The proportion of concept mentions decreases with increasing dataset size. Correspondingly, validation text context counts stay constant while an increase in training data decreases available test data.

Triple Counts



Text Counts



The bar plots above report the absolute counts of triples (called *Vertex Triples* in the former table) and associated text. The resulting triple split behaves correspondingly to the mention split with a proportional behaviour between training/test data. The overall triple count more than doubles between T and L. Associated text data increases with respect to the increase in closed-world mentions.

The following table shows all relations present in CoDEX ordered by ratio as described in 5. On the right-hand side, it is detailed which relations are present in which dataset variant. The respective right side shows whether it is selected to remain in the dataset. The left side marks whether the relation was selected to pick concept entities.

8.2. Hyperparameters

All models are trained on Nvidia GTX 2080TI or RTX A6000 graphics cards using PyTorch version 1.11. KGC models use the 1.8 implementation of PyKEEN and the text encoder uses Huggingface’s transformer implementation 4.19. The amount of text samples per epoch varies between different model configurations and is a combination of “max contexts” (the total amount of text contexts associated with an entity during training) “max contexts per sample” (the number of text contexts used by a model per training step). Model training is stopped if the performance did not increase on the target metric (i.e. hits@10) on the validation split by more than 0.001 for a few epochs. In the following, the hyperparameters for the models presented in 6 are enumerated.

	Ranking				Linking			
	Tiny	Small	Medium	Large	Tiny	Small	Medium	Large
Embedding Dims.	800	500	800	200	200	800	800	200
Unfrozen Layer	5	0	11	0	5	11	11	0
Regularizer Weight	7.16e-1	7.73e-1	7.12e-2	5.86e-3	6.56e-1	9.02e-3	7.12e-2	5.86e-3
Contexts per Sample	1	1	1	1	1	1	1	1
Maximum Contexts	10	100	10	10	10	10	10	10
Masked	false	false	false	false	true	true	false	false
Batch Size	4	8	40	20	4	8	40	20
Learning Rate	3.94e-6	4.93e-6	4.39e-6	8.99e-6	8.47e-6	6.67e-6	4.39e-6	8.99e-6
Weight Decay	3.48e-2	1.19e-2	6.64e-4	1.49e-4	1.22e-4	2.16e-4	6.64e-4	1.49e-4
Seed	5629275	4828059	2773483	4076657	8697782	9387603	2773483	4076657

Table 3

JOINT (single context) - best model selected after random search.

	Ranking				Linking			
	Tiny	Small	Medium	Large	Tiny	Small	Medium	Large
Embedding Dims.	800	800	800	800	800	800	800	800
Unfrozen Layer	1	1	1	1	1	1	1	1
Regularizer Weight	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Contexts per Sample	40	40	40	10	40	40	40	10
Maximum Contexts	100	100	1000	1000	100	100	1000	1000
Masked	false	false	false	false	false	false	false	false
Batch Size	1	1	1	4	1	1	1	4
Subbatch Size	5	5	10	10	5	5	10	10
Learning Rate	5.00e-6	5.00e-6	5.00e-6	5.00e-6	5.00e-6	5.00e-6	5.00e-6	5.00e-6
Weight Decay	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Seed	4733486	5969007	7460979	6929361	4733486	5969007	7460979	6929361

Table 4

JOINT (multi-context) - best model selected after grid search.

	Ranking				Linking			
	Tiny	Small	Medium	Large	Tiny	Small	Medium	Large
Contexts per Sample	1	1	1	1	1	1	1	1
Maximum Contexts	1	1	1	1	1	1	1	1
Masked	false	false	false	false	false	false	false	false
Batch Size	10	10	10	10	10	10	10	10
Subbatch Size	10	10	10	10	10	10	10	10
Learning Rate	5.00e-5	5.00e-5	5.00e-5	5.00e-5	5.00e-5	5.00e-5	5.00e-5	5.00e-5
Seed	1705119	2298300	3899079	8847385	1705119	2298300	3899079	8847385

Table 5
OWE (single context) - best model selected after grid search.

	Ranking				Linking			
	Tiny	Small	Medium	Large	Tiny	Small	Medium	Large
Contexts per Sample	10	10	10	10	10	10	10	10
Maximum Contexts	100	100	100	100	100	100	100	100
Masked	false	false	false	false	true	false	true	false
Batch Size	1	1	1	1	1	1	1	1
Subbatch Size	10	10	10	10	10	10	10	10
Learning Rate	1.00e-6	1.00e-6	1.00e-6	1.00e-6	1.00e-6	1.00e-6	1.00e-6	1.00e-6
Seed	3443300	3250085	4051229	6248903	9294686	9773709	9790717	6248903

Table 6
OWE (multi-context) - best model selected after grid search.

	Complex			
	Tiny	Small	Medium	Large
Embedding Dims.	300	300	300	500
Learning Rate	1	0.6	0.4	0.6
Regularizer Weight	0.3	0.1	0.1	0.1
Batch Size	64	64	64	64
Seed	1174584270	2593575093	2203942071	2649116927

Table 7
OWE reference embeddings - best model selected after grid search.

9. Bibliography

- [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, Synthesis Lectures on Data, Semantics, and Knowledge 12 (2021) 1–257.
- [2] R. Hoehndorf, P. N. Schofield, G. V. Gkoutos, The role of ontologies in biological and biomedical research: a functional perspective, Briefings in Bioinformatics 16 (2015) 1069–1080. URL: <https://doi.org/10.1093/bib/bbv011>. doi:10.1093/bib/bbv011. arXiv:<https://academic.oup.com/bib/article-pdf/16/6/1069/607091/bbv011.pdf>.
- [3] M. Ali, M. Berrendorf, M. Galkin, V. Thost, T. Ma, V. Tresp, J. Lehmann, Improving inductive link prediction using hyper-relational facts, in: International Semantic Web Conference, Springer, 2021, pp. 74–92.

- [4] B. Shi, T. Weninger, Open-world knowledge graph completion, arXiv preprint arXiv:1711.03438 (2017).
- [5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, ACM, New York, NY, USA, 2008, pp. 1247–1250. URL: <http://doi.acm.org/10.1145/1376616.1376746>. doi:10.1145/1376616.1376746.
- [6] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.
- [7] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data., in: ICML, volume 11, 2011, pp. 809–816.
- [8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in neural information processing systems, 2013, pp. 2787–2795.
- [9] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint arXiv:1412.6575 (2014).
- [10] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, 2016, pp. 2071–2080.
- [11] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
- [12] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).
- [13] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, arXiv preprint arXiv:1906.01195 (2019).
- [14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.
- [15] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information. corr abs/1607.04606, URL <http://arxiv.org/abs/1607.04606> (2016).
- [16] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
- [17] R. Socher, D. Chen, C. D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in neural information processing systems, 2013, pp. 926–934.
- [18] R. Xie, Z. Liu, J. Jia, H. Luan, M. Sun, Representation learning of knowledge graphs with entity descriptions, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
- [19] C. Fu, Z. Li, Q. Yang, Z. Chen, J. Fang, P. Zhao, J. Xu, Multiple interaction attention model for open-world knowledge graph completion, in: International Conference on Web Information Systems Engineering, Springer, 2020, pp. 630–644.
- [20] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, J. Tang, Kepler: A unified model for knowledge embedding and pre-trained language representation, Transactions of the Association for Computational Linguistics 9 (2021) 176–194.
- [21] H. Shah, J. Villmow, A. Ulges, U. Schwanecke, F. Shafait, An open-world extension to knowledge graph completion models, in: Thirty-Third AAAI Conference on Artificial

Intelligence, 2019.

- [22] H. Shah, J. Villmow, A. Ulges, Relation specific transformations for open world knowledge graph completion, in: Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs), Association for Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 79–84. URL: <https://www.aclweb.org/anthology/2020.textgraphs-1.9>.
- [23] Y. Zhou, S. Shi, H. Huang, Weighted aggregator for the open-world knowledge graph completion, in: International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2020, pp. 283–291.
- [24] D. Daza, M. Cochez, P. Groth, Inductive entity representations from text via link prediction, in: Proceedings of the Web Conference 2021, 2021, pp. 798–808.
- [25] 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, pp. 722–735.
- [26] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, Association for Computational Linguistics, Beijing, China, 2015, pp. 57–66. URL: <https://www.aclweb.org/anthology/W15-4007>. doi:10.18653/v1/W15-4007.
- [27] T. Safavi, D. Koutra, CoDEX: A Comprehensive Knowledge Graph Completion Benchmark, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 8328–8350.
- [28] F. Hamann, A. Ulges, D. Krechel, R. Bergmann, Open-world knowledge graph completion benchmarks for knowledge discovery, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2021, pp. 252–264.
- [29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.
- [30] S. E. Robertson, S. Walker, Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval, in: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Springer-Verlag New York, Inc., 1994, pp. 232–241.
- [31] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization., *Journal of machine learning research* 12 (2011).
- [32] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann, Pykeen 1.0: A python library for training and evaluating knowledge graph embeddings, *arXiv preprint arXiv:2007.14175* (2020).
- [33] 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, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.
- [34] W. Falcon, The PyTorch Lightning team, PyTorch Lightning, 2019. URL: <https://github.com/Lightning-AI/lightning>. doi:10.5281/zenodo.3828935.

- [35] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
- [36] B. Mitra, N. Craswell, Neural models for information retrieval, arXiv preprint arXiv:1705.01509 (2017).
- [37] J. Lin, R. Nogueira, A. Yates, Pretrained transformers for text ranking: Bert and beyond, Synthesis Lectures on Human Language Technologies 14 (2021) 1–325.