CoDEx: A Comprehensive Knowledge Graph Completion Benchmark

Tara Safavi
University of Michigan
tsafavi@umich.edu

Danai Koutra
University of Michigan
dkoutra@umich.edu

Abstract

We present CoDEx, a set of knowledge graph **CO**mpletion **D**atasets **EX**tracted from Wikidata and Wikipedia that improve upon existing knowledge graph completion benchmarks in scope and level of difficulty. In terms of scope, CoDEx comprises three knowledge graphs varying in size and structure, multilingual descriptions of entities and relations, and tens of thousands of *hard negative* triples that are plausible but verified to be false. To characterize CoDEx, we contribute thorough empirical analyses and benchmarking experiments. First, we analyze each CoDEx dataset in terms of logical relation patterns. Next, we report baseline link prediction and triple classification results on CoDEx for five extensively tuned embedding models. Finally, we differentiate CoDEx from the popular FB15K-237 knowledge graph completion dataset by showing that CoDEx covers more diverse and interpretable content, and is a more difficult link prediction benchmark. Data, code, and pretrained models are available at https://bit.ly/2EPbrJs.

1 Introduction

Knowledge graphs are multi-relational graphs that express facts about the world by connecting entities (people, places, things, concepts) via different types of relationships. The field of automatic knowledge graph completion (**KGC**), which is motivated by the fact that knowledge graphs are usually incomplete, is an active research direction spanning several subfields of artificial intelligence (Nickel et al., 2015; Wang et al., 2017; Ji et al., 2020).

As progress in artificial intelligence depends heavily on data, a relevant and high-quality benchmark is imperative to evaluating and advancing the state of the art in KGC. However, the field has largely remained static in this regard over the past decade. Outdated subsets of Freebase (Bollacker et al., 2008) are most commonly used for evaluation in KGC, even though Freebase had known quality issues (Tanon et al., 2016) and was eventually deprecated in favor of the more recent Wikidata knowledge base (Vrandečić and Krötzsch, 2014).

Indeed, KGC benchmarks extracted from Freebase like FB15K and FB15K-237 (Bordes et al., 2013; Toutanova and Chen, 2015) are questionable in quality. For example, FB15K was shown to have train/test leakage (Toutanova and Chen, 2015). Later in this paper (§ 6.2), we will show that a relatively large proportion of relations in FB15K-237 can be covered by a trivial frequency rule.

To address the need for a solid benchmark in KGC, we present **CoDEx**, a set of knowledge graph **CO**mpletion **D**atasets **EX**tracted from Wikidata and its sister project Wikipedia. Inasmuch as Wikidata is considered the successor of Freebase, CoDEx improves upon existing Freebase-based KGC benchmarks in terms of scope and level of difficulty (Table 1). Our contributions include:

Foundations We survey evaluation datasets in encyclopedic knowledge graph completion to motivate a new benchmark (§ 2 and Appendix A).

Data We introduce CoDEx, a benchmark consisting of three knowledge graphs varying in size and structure, entity types, multilingual labels and descriptions, and—unique to CoDEx—manually verified *hard negative* triples (§ 3). To better understand CoDEx, we analyze the logical relation patterns in each of its datasets (§ 4).

Benchmarking We conduct large-scale model selection and benchmarking experiments, reporting baseline link prediction and triple classification results on CoDEx for five widely used embedding models from different architectural classes (§ 5).

Comparative analysis Finally, to demonstrate the unique value of CoDEx, we differentiate

Table 1: Qualitative comparison of CoDEx datasets to existing Freebase-based KGC datasets (§ 2.1).

	Freebase variants (FB15K, FB15K-237)	CoDEx datasets
Scope (domains)	Multi-domain, with a strong focus on awards, entertainment, and sports (§ 6.1 and Appendix E)	Multi-domain, with focuses on writing, entertainment, music, politics, journalism, academics, and science (§ 6.1 and Appendix E)
Scope (auxiliary data)	Various decentralized versions of FB15K with, e.g., entity types (Xie et al., 2016), sampled negatives (Socher et al., 2013), and more (Table 8)	Centralized repository of three datasets with entity types, multilingual text, and manually annotated hard negatives (§ 3)
Level of difficulty	FB15K has severe train/test leakage from inverse relations (Toutanova and Chen, 2015); while removal of inverse relations makes FB15K-237 harder than FB15K, FB15K-237 still has a high proportion of easy-to-predict relational patterns (§ 6.2)	Inverse relations removed from all datasets to avoid train/test leakage (§ 3.2); manually annotated hard negatives for the task of triple classification (§ 3.4); few trivial patterns for the task of link prediction (§ 6.2)

CoDEx from FB15K-237 in terms of both content and difficulty (§ 6). We show that CoDEx covers more diverse and interpretable content, and is a more challenging link prediction benchmark.

2 Existing datasets

We begin by surveying existing KGC benchmarks. Table 8 in Appendix A provides an overview of evaluation datasets and tasks on a *per-paper* basis across the artificial intelligence, machine learning, and natural language processing communities.

Note that we focus on *data* rather than *models*, so we only overview relevant evaluation benchmarks here. For more on existing KGC models, both neural and symbolic, we refer the reader to (Meilicke et al., 2018) and (Ji et al., 2020).

2.1 Freebase extracts

These datasets, extracted from the Freebase knowledge graph (Bollacker et al., 2008), are the most popular for KGC (see Table 8 in Appendix A).

FB15K was introduced by Bordes et al. (2013). It contains 14,951 entities, 1,345 relations, and 592,213 triples covering several domains, with a strong focus on awards, entertainment, and sports.

FB15K-237 was introduced by Toutanova and Chen (2015) to remedy data leakage in FB15K, which contains many test triples that invert triples in the training set. FB15K-237 contains 14,541 entities, 237 relations, and 310,116 triples. We compare FB15K-237 to CoDEx in § 6 to assess each dataset's content and relative difficulty.

2.2 Other encyclopedic datasets

NELL-995 (Xiong et al., 2017) was taken from the Never Ending Language Learner (NELL) system (Mitchell et al., 2018), which continuously reads the web to obtain and update its knowledge. NELL-995, a subset of the 995th iteration of NELL, contains 75,492 entities, 200 relations, and 154,213 triples. While NELL-995 is general and covers many domains, its mean average precision was less than 50% around its 1000th iteration (Mitchell et al., 2018). A cursory inspection reveals that many of the triples in NELL-995 are nonsensical or overly generic, suggesting that NELL-995 is not a meaningful dataset for KGC evaluation.[1]

YAGO3-10 (Dettmers et al., 2018) is a subset of YAGO3 (Mahdisoltani et al., 2014), which covers portions of Wikipedia, Wikidata, and WordNet. YAGO3-10 has 123,182 entities, 37 relations, and 1,089,040 triples mostly limited to facts about people and locations. While YAGO3-10 is a high-precision dataset, it was recently shown to be too easy for link prediction because it contains a large proportion of duplicate relations (Akrami et al., 2020; Pezeshkpour et al., 2020).

2.3 Domain-specific datasets

In addition to large encyclopedic knowledge graphs, it is common to evaluate KGC methods on at least one smaller, domain-specific dataset, typically drawn from the **WordNet** semantic network (Miller, 1998; Bordes et al., 2013). Other choices include the Unified Medical Language System (**UMLS**) database (McCray, 2003), the **Alyawarra kinship** dataset (Kemp et al., 2006), the **Countries** dataset (Bouchard et al., 2015), and variants of a synthetic "**family tree**" (Hinton, 1986). As our focus in this paper is encyclopedic knowledge, we do not cover these datasets further.

[1]Some examples: (*politician:jobs*, *worksfor*, *county:god*), (*person:buddha001*, *parentofperson*, *person:jesus*)

Table 2: CoDEx datasets. (+): Positive (*true*) triples. (-): Verified negative (*false*) triples (§ 3.4). We compute multilingual coverage over all labels, descriptions, and entity Wikipedia extracts successfully retrieved for the respective dataset in Arabic (ar), German (de), English (en), Spanish (es), Russian (ru), and Chinese (zh).

	$\|E\|$	$\|R\|$	Triples $E \times R \times E$					Multilingual coverage					
			Train (+)	Valid (+)	Test (+)	Valid (-)	Test (-)	ar	de	en	es	ru	zh
CoDEx-S	2,034	42	32,888	1827	1828	1827	1828	77.38	91.87	96.38	91.55	89.17	79.36
CoDEx-M	17,050	51	185,584	10,310	10,311	10,310	10,311	75.80	95.20	96.95	87.91	81.88	69.63
CoDEx-L	77,951	69	551,193	30,622	30,622	-	-	67.47	90.84	92.40	81.30	71.12	61.06

3 Data collection

In this section we describe the pipeline used to construct CoDEx. For reference, we define a knowledge graph G as a multi-relational graph consisting of a set of entities E, relations R, and factual statements in the form of (*head*, *relation*, *tail*) triples $(h, r, t) \in E \times R \times E$.

3.1 Seeding the collection

We collected an initial set of triples using a type of snowball sampling (Goodman, 1961). We first manually defined a broad seed set of entity and relation types common to 13 domains: Business, geography, literature, media and entertainment, medicine, music, news, politics, religion, science, sports, travel, and visual art. Examples of seed entity types include *airline*, *journalist*, and *religious text*; corresponding seed relation types in each respective domain include *airline alliance*, *notable works*, and *language of work or name*. Table 9 in Appendix B gives all seed entity and relation types.

Using these seeds, we retrieved an initial set of 380,038 entities, 75 relations, and 1,156,222 triples by querying Wikidata for statements of the form (*head entity of seed type*, *seed relation type*, ?).

3.2 Filtering the collection

To create smaller data snapshots, we filtered the initial 1.15 million triples to k-cores, which are maximal subgraphs G' of a given graph G such that every node in G' has a degree of at least k (Batagelj and Zaveršnik, 2011).[2] We constructed three CoDEx datasets (Table 2):

- **CoDEx-S** (k = 15), which has 36k triples. Because of its smaller size, we recommend that CoDEx-S be used for model testing and debugging, as well as evaluation of methods that are less computationally efficient (e.g., symbolic search-based approaches).
- **CoDEx-M** (k = 10), which has 206k triples. CoDEx-M is all-purpose, being comparable in size to FB15K-237 (§ 2.1), one of the most popular benchmarks for KGC evaluation.
- **CoDEx-L** (k = 5), which has 612k triples. CoDEx-L is comparable in size to FB15K (§ 2.1), and can be used for both general evaluation and "few-shot" evaluation.

We also release the raw dump that we collected via snowball sampling, but focus on CoDEx-S through L for the remainder of this paper.

To minimize train/test leakage, we removed inverse relations from each dataset (Toutanova and Chen, 2015). We computed (*head*, *tail*) and (*tail*, *head*) overlap between all pairs of relations, and removed each relation whose entity pair set overlapped with that of another relation more than 50% of the time. Finally, we split each dataset into 90/5/5 train/validation/test triples such that the validation and test sets contained only entities and relations seen in the respective training sets.

3.3 Auxiliary information

An advantage of Wikidata is that it links entities and relations to various sources of rich auxiliary information. To enable tasks that involve joint learning over knowledge graph structure and such additional information, we collected:

- **Entity types** for each entity as given by Wikidata's *instance of* and *subclass of* relations;
- **Wikidata labels and descriptions** for entities, relations, and entity types; and
- **Wikipedia page extracts** (introduction sections) for entities and entity types.

For the latter two, we collected text where available in Arabic, German, English, Spanish, Russian, and Chinese. We chose these languages because they are all relatively well-represented on Wikidata (Kaffee et al., 2017). Table 2 provides the coverage by language for each CoDEx dataset.

[2] A similar approach was used to extract the FB15K dataset from Freebase (Bordes et al., 2013).

Table 3: Selected examples of hard negatives in CoDEx with explanations.

Negative	Explanation
(*Frédéric Chopin*, *occupation*, *conductor*)	Chopin was a pianist and a composer, not a conductor.
(*Lesotho*, *official language*, *American English*)	English, not American English, is an official language of Lesotho.
(*Senegal*, *part of*, *Middle East*)	Senegal is part of West Africa.
(*Simone de Beauvoir*, *field of work*, *astronomy*)	Simone de Beauvoir's field of work was primarily philosophy.
(*Vatican City*, *member of*, *UNESCO*)	Vatican City is a UNESCO World Heritage Site but not a member state.

3.4 Hard negatives for evaluation

Knowledge graphs are unique in that they only contain positive statements, meaning that triples *not* observed in a given knowledge graph are not necessarily false, but merely unseen; this is called the Open World Assumption (Galárraga et al., 2013). However, most machine learning tasks on knowledge graphs require negatives in some capacity. While different negative sampling strategies exist (Cai and Wang, 2018), the most common approach is to randomly perturb observed triples to generate negatives, following Bordes et al. (2013).

While random negative sampling is beneficial and even necessary in the case where a large number of negatives is needed (i.e., training), it is not necessarily useful for evaluation. For example, in the task of triple classification, the goal is to discriminate between positive (true) and negative (false) triples. As we show in § 5.5, triple classification over randomly generated negatives is trivially easy for state-of-the-art models because random negatives are generally not meaningful or plausible. Therefore, we generate and manually evaluate *hard negatives* for KGC evaluation.

Generation To generate hard negatives, we used each pre-trained embedding model from § 5.2 to predict tail entities of triples in CoDEx. For each model, we took as candidate negatives the triples $(h, r, \hat{t})$ for which (i) the type of the predicted tail entity $\hat{t}$ matched the type of the true tail entity t; (ii) $\hat{t}$ was ranked in the top-10 predictions by that model; and (iii) $(h, r, \hat{t})$ was not observed in G.

Annotation We manually labeled all candidate negative triples generated for CoDEx-S and CoDEx-M as *true* or *false* using the guidelines provided in Appendix C.[3] We randomly selected among the triples labeled as *false* to create **validation and test negatives for CoDEx-S and CoDEx-M**, examples of which are given in Table 3. To assess the quality of our annotations, we gathered judgments from two independent native English speakers on a random selection of 100 candidate negatives. The annotators were provided the instructions from Appendix C. On average, our labels agreed with those of the annotators 89.5% of the time. Among the disagreements, 81% of the time we assigned the label *true* whereas the annotator assigned the label *false*, meaning that we were comparatively conservative in labeling negatives.

[3]We are currently investigating methods for obtaining high-quality crowdsourced annotations of negatives for CoDEx-L.

4 Analysis of relation patterns

To give an idea of the types of reasoning necessary for models to perform well on CoDEx, we analyze the presence of learnable binary relation patterns within CoDEx. The three main types of such patterns in knowledge graphs are **symmetry**, **inversion**, and **compositionality** (Trouillon et al., 2019; Sun et al., 2019). We address symmetry and compositionality here, and omit inversion because we specifically removed inverse relations to avoid train/test leakage (§ 3.2).

4.1 Symmetry

Symmetric relations are relations r for which $(h, r, t) \in G$ implies $(t, r, h) \in G$. For each relation, we compute the number of its (*head*, *tail*) pairs that overlap with its (*tail*, *head*) pairs, divided by the total number of pairs, and take those with 50% overlap or higher as symmetric. CoDEx datasets have five such relations: *diplomatic relation*, *shares border with*, *sibling*, *spouse*, and *unmarried partner*. Table 4 gives the proportion of triples containing symmetric relations per dataset. Symmetric patterns are more prevalent in CoDEx-S, whereas the larger datasets are mostly **antisymmetric**, i.e., $(h, r, t) \in G$ implies $(t, r, h) \notin G$.

4.2 Composition

Compositionality captures **path rules** of the form $(h, r_1, x_1), \ldots, (x_n, r_n, t) \rightarrow (h, r, t)$. To learn these rules, models must be capable of "multi-hop" reasoning on knowledge graphs (Guu et al., 2015).

Table 4: Relation patterns in CoDEx. For symmetry, we give the proportion of triples containing a symmetric relation. For composition, we give the proportion of triples participating in a rule of length two or three.

	CoDEx-S	CoDEx-M	CoDEx-L
Symmetry	17.46%	4.01%	3.29%
Composition	10.09%	16.55%	31.84%

To identify compositional paths, we use the AMIE3 system (Lajus et al., 2020), which outputs rules with confidence scores that capture how many times those rules are seen versus violated, to identify paths of lengths two and three; we omit longer paths as they are relatively costly to compute. We identify 26, 44, and 93 rules in CoDEx-S, CoDEx-M, and CoDEx-L, respectively, with average confidence (out of 1) of 0.630, 0.556, and 0.459. Table 4 gives the percentage of triples per dataset participating in a discovered rule.

Evidently, composition is especially prevalent in CoDEx-L. An example rule in CoDEx-L is "if *X* was founded by *Y*, and *Y*'s country of citizenship is *Z*, then the country [i.e., of origin] of *X* is *Z*" (confidence 0.709). We release these rules as part of CoDEx for further development of KGC methodologies that incorporate or learn rules.

5 Benchmarking

Next, we benchmark performance on CoDEx for the tasks of link prediction and triple classification. To ensure that models are fairly and accurately compared, we follow Ruffinelli et al. (2020), who conducted what is (to the best of our knowledge) the largest-scale hyperparameter tuning study of knowledge graph embeddings to date.

Note that CoDEx can be used to evaluate any type of KGC method. However, we focus on embeddings in this section due to their widespread usage in modern NLP (Ji et al., 2020).

5.1 Tasks

Link prediction The link prediction task is conducted as follows: Given a test triple (h, r, t), we construct queries $(?, r, t)$ and $(h, r, ?)$. For each query, a model scores candidate head (tail) entities $\hat{h}$ $(\hat{t})$ according to its belief that $\hat{h}$ $(\hat{t})$ completes the triple (i.e., answers the query). The goal is of link prediction is to rank true triples $(\hat{h}, r, t)$ or $(h, r, \hat{t})$ higher than false and unseen triples.

Link prediction performance is evaluated with mean reciprocal rank (**MRR**) and **hits@**k. MRR is the average reciprocal of each ground-truth entity's rank over all $(?, r, t)$ and $(h, r, ?)$ test triples. Hits@k measures the proportion of test triples for which the ground-truth entity is ranked in the top-k predicted entities. In computing these metrics, we exclude the predicted entities for which $(\hat{h}, r, t) \in G$ or $(h, r, \hat{t}) \in G$ so that known positive triples do not artificially lower ranking scores. This is called "filtering" (Bordes et al., 2013).

Triple classification Given a triple (h, r, t), the goal of triple classification is to predict a corresponding label $y \in \{-1, 1\}$. Since knowledge graph embedding models output real-valued scores for triples, we convert these scores into labels by selecting a decision threshold per relation on the validation set such that validation accuracy is maximized for the model in question. A similar approach was used by Socher et al. (2013).

We compare results on three sets of evaluation negatives: (1) We generate one negative per positive by replacing the positive triple's tail entity by a tail entity t' sampled **uniformly at random**; (2) We generate negatives by sampling tail entities according to their **relative frequency in the tail slot** of all triples; and (3) We use the CoDEx **hard negatives**. We measure **accuracy** and **F1 score**.

5.2 Models

We compare the following embedding methods: **RESCAL** (Nickel et al., 2011), **TransE** (Bordes et al., 2013), **ComplEx** (Trouillon et al., 2016), **ConvE** (Dettmers et al., 2018), and **TuckER** (Balazevic et al., 2019b). These models represent several classes of architecture, from linear (RESCAL, TuckER, ComplEx) to translational (TransE) to nonlinear/learned (ConvE). Appendix D provides more specifics on each model.

5.3 Model selection

As recent studies have observed that training strategies are equally, if not more, important than architecture for link prediction (Kadlec et al., 2017; Lacroix et al., 2018; Ruffinelli et al., 2020), we search across a large range of hyperparameters to ensure a truly fair comparison. To this end we use the PyTorch-based LibKGE framework for training and selecting knowledge graph embeddings.[4] In the remainder of this section we outline the most important parameters of our model selection process.

[4]https://github.com/uma-pi1/kge

Table 5: Comparison of link prediction performance on CoDEx.

	CoDEx-S			CoDEx-M			CoDEx-L		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
RESCAL	0.404	0.293	0.623	0.317	0.244	0.456	0.304	0.242	0.419
TransE	0.354	0.219	0.634	0.303	0.223	0.454	0.187	0.116	0.317
ComplEx	**0.465**	**0.372**	**0.646**	**0.337**	**0.262**	**0.476**	0.294	0.237	0.400
ConvE	0.444	0.343	0.635	0.318	0.239	0.464	0.303	0.240	0.420
TuckER	0.444	0.339	0.638	0.328	0.259	0.458	**0.309**	**0.244**	**0.430**

Table 10 in Appendix F gives further details and all hyperparameter ranges and values. All experiments were run on a single NVIDIA Tesla V100 GPU with 16 GB of RAM.

Training negatives Given a set of positive training triples $\{(h, r, t)\}$, we compare three types of negative sampling strategy implemented by LibKGE: (a) **NegSamp**, or randomly corrupting head entities h or tail entities t to create negatives; (b) **1vsAll**, or treating *all* possible head/tail corruptions of (h, r, t) as negatives, including the corruptions that are actually positives; and (c) **KvsAll**, or treating batches of head/tail corruptions *not* seen in the knowledge graph as negatives.

Loss functions We consider the following loss functions: (i) **MR** or margin ranking, which aims to maximize a margin between positive and negative triples; (ii) **BCE** or binary cross-entropy, which is computed by applying the logistic sigmoid to triple scores; and (iii) **CE** or cross-entropy between the softmax over the entire distribution of triple scores and the label distribution over all triples, normalized to sum to one.

Search strategies We select models using the Ax platform, which supports hyperparameter search using both quasi-random sequences of generated configurations and Bayesian optimization (BO) with Gaussian processes.[5] At a high level, for each dataset and model, we generate both quasi-random and BO trials *per negative sampling and loss function combination*, ensuring that we search over a wide range of hyperparameters for different types of training strategy. Appendix F provides specific details on the search strategy for each dataset, which was determined according to resource constraints and observed performance patterns.

[5]https://ax.dev/

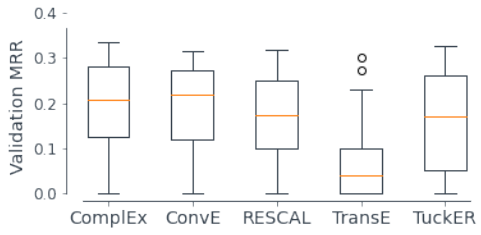

Figure 1: Distribution of validation MRR, CoDEx-M.

5.4 Link prediction results

Table 5 gives link prediction results. We find that ComplEx is the **best at modeling symmetry and antisymmetry**, and indeed it was designed specifically to improve upon bilinear models that do not capture symmetry, like DistMult (Trouillon et al., 2016). As such, it performs the best on CoDEx-S, which has the highest proportion of symmetric relations. For example, on the most frequent symmetric relation (*diplomatic relation*), ComplEx achieves 0.859 MRR, compared to 0.793 for ConvE, 0.490 for RESCAL, and 0.281 for TransE.

By contrast, TuckER is **strongest at modeling compositional relations**, so it performs best on CoDEx-L, which has a high degree of compositionality. For example, on the most frequent compositional relation in CoDEx-L (*languages spoken, written, or signed*), TuckER achieves 0.465 MRR, compared to 0.464 for RESCAL, 0.463 for ConvE, 0.456 for ComplEx, and 0.385 for TransE. By contrast, since CoDEx-M is mostly asymmetric and non-compositional, ComplEx performs best because of its ability to model asymmetry.

Effect of hyperparameters As shown by Figure 1, hyperparameters have a strong impact on link prediction performance: Validation MRR for all models varies by *over 30 percentage points* depending on the training strategy and input configuration. This finding is consistent with previous observations in the literature (Kadlec et al., 2017; Ruffinelli et al., 2020). Appendix F provides the best configurations for each model.

Table 6: Comparison of triple classification performance on CoDEx by negative generation strategy.

	CoDEx-S						CoDEx-M					
	Uniform		Relative freq.		Hard neg.		Uniform		Relative freq.		Hard neg.	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
RESCAL	0.972	0.972	0.916	0.920	**0.843**	**0.852**	0.977	0.976	0.921	0.922	0.818	0.815
TransE	0.974	0.974	0.919	0.923	0.829	0.837	**0.986**	**0.986**	0.932	0.933	0.797	0.803
ComplEx	**0.975**	**0.975**	**0.927**	**0.930**	0.836	0.846	0.984	0.984	0.930	0.933	0.824	0.818
ConvE	0.972	0.972	0.921	0.924	0.841	0.846	0.979	0.979	**0.934**	**0.935**	**0.826**	**0.829**
TuckER	0.973	0.973	0.917	0.920	0.840	0.846	0.977	0.977	0.920	0.922	0.823	0.816

Overall, we find that the choice of loss function in particular significantly impacts model performance. Each model consistently **achieved its respective peak performance with cross-entropy (CE) loss**, a finding which is corroborated by several other KGC comparison papers (Kadlec et al., 2017; Ruffinelli et al., 2020; Jain et al., 2020). As far as negative sampling techniques, we do not find that a single strategy is dominant, suggesting that the choice of loss function is more important.

5.5 Triple classification results

Table 6 gives triple classification results. Evidently, triple classification on ***randomly*** **generated negatives is a nearly-solved task**. On negatives generated uniformly at random, performance scores are nearly identical at almost 100% accuracy. Even with a negative sampling strategy "smarter" than uniform random, all models perform well.

Hard negatives Classification performance degenerates considerably on our hard negatives, around 8 to 11 percentage points from relative frequency-based sampling and 13 to 19 percentage points from uniformly random sampling. Relative performance also varies: In contrast to our link prediction task in which ComplEx and TuckER were by far the strongest models, RESCAL is slightly stronger on the CoDEx-S hard negatives, whereas ConvE performs best on the CoDEx-M hard negatives. These results indicate that **triple classification is indeed a distinct task** that requires different architectures and, in many cases, different training strategies (Appendix F).

We believe that few recent works use triple classification as an evaluation task because of the lack of true hard negatives in existing benchmarks. Early works reported high triple classification accuracy on sampled negatives (Socher et al., 2013; Wang et al., 2014), perhaps leading the community to believe that the task was nearly solved. However, our results demonstrate that the task is far from solved when the negatives are plausible but truly false.

6 Comparative case study

Finally, we conduct a comparative analysis between CoDEx-M and FB15K-237 (§ 2.1) to demonstrate the unique value of CoDEx. We choose FB15K-237 because it is the most popular encyclopedic KGC benchmark after FB15K, which was already shown to be an easy dataset by Toutanova and Chen (2015). We choose CoDEx-M because it is the closest in size to FB15K-237.

6.1 Content

We first compare the content in CoDEx-M, which is extracted from Wikidata, with that of FB15K-237, which is extracted from Freebase. For brevity, Figure 2 compares the top-15 relations by mention count in the two datasets. Appendix E provides more content comparisons.

Diversity The most common relation in CoDEx-M is *occupation*, which is because most people on Wikidata have multiple occupations listed. By contrast, the frequent relations in FB15K-237 are mostly related to awards and film. In fact, over 25% of all triples in FB15K-237 belong to the */award* relation domain, suggesting that CoDEx covers a more diverse selection of content.

Interpretability The Freebase-style relations are also arguably less interpretable than those in Wikidata. Whereas Wikidata relations have concise natural language labels, the Freebase relation labels are hierarchical, often at five or six levels of hierarchy (Figure 2). Moreover, all relations in Wikidata are binary, whereas some Freebase relations are n-nary (Tanon et al., 2016), meaning that they connect more than two entities. The relations containing a dot (".") are such n-nary relations, and are difficult to reason about without understanding the structure of Freebase, which has been deprecated.

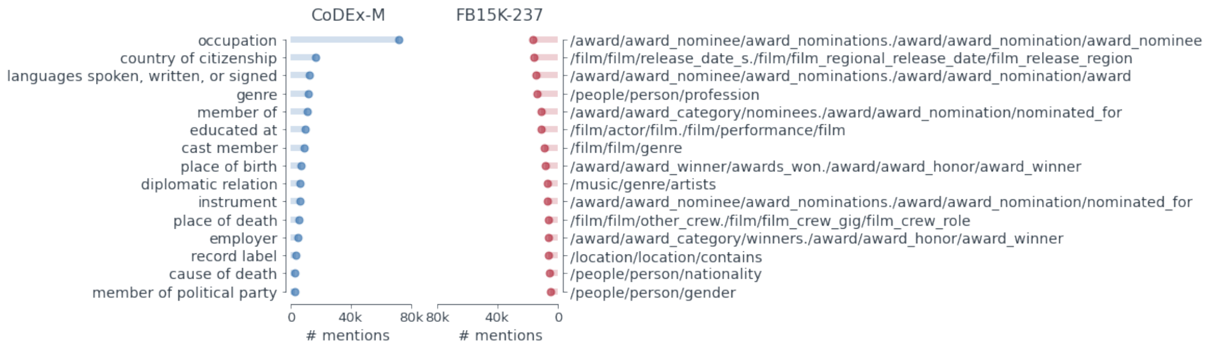

Figure 2: Top-15 most frequent relations in CODEX-M and FB15K-237.

We further discuss the impact of such n-nary relations for link prediction in the following section.

6.2 Difficulty

Next, we compare the datasets in a link prediction task to show that CODEX-M is more difficult.

Baseline We devise a "non-learning" link prediction baseline. Let $(h, r, ?)$ be a test query. Our baseline scores candidate tail entities by their relative frequency in the tail slot of all training triples mentioning r, filtering out tail entities t for which (h, r, t) is already observed in the training set. If all tail entities t are filtered out, we score entities by frequency before filtering. The logic of our approach works in reverse for $(?, r, t)$ queries. In evaluating our baseline, we follow LibKGE's protocol for breaking ties in ranking (i.e., for entities that appear with equal frequency) by taking the mean rank of all entities with the same score.

Setup We compare our baseline to the best pretrained embedding model per dataset: RESCAL for FB15K-237, which was released by Ruffinelli et al. (2020), and ComplEx for CODEX-M. We evaluate performance with MRR and Hits@10. Beyond overall performance, we also compute *per-relation improvement* of the respective embedding over our baseline in terms of percentage points MRR. This measures the amount of learning beyond frequency statistics necessary for each relation.

Results and discussion Table 7 compares the overall performance of our baseline versus the best embedding per dataset, and Figure 3 shows the improvement of the respective embedding over our baseline per relation type on each dataset. The improvement of the embedding is much smaller on FB15K-237 than CODEX-M, and in fact our baseline performs on par with or even *outperforms* the embedding on FB15K-237 for some relation types.

Table 7: Overall performance (MRR) of our frequency baseline versus the best embedding nodel per benchmark. "Improvement" refers to the improvement of the embedding over the baseline.

	Baseline	Embedding	Improvement
FB15K-237	0.236	0.356	+0.120
CODEX-M	0.135	0.337	+0.202

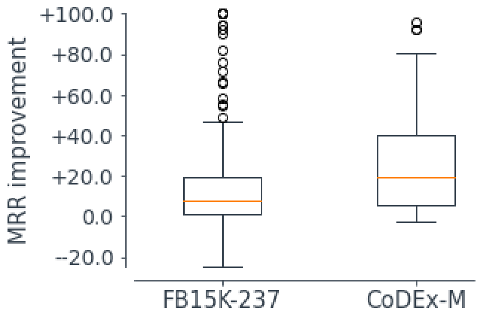

Figure 3: Improvement in MRR of the embedding over our frequency baseline per relation type. Negative means that our baseline outperforms the embedding. The medians are 8.27 and 20.04 percentage points on FB15K-237 and CODEX-M, respectively.

To further explore these cases, Figure 4 gives the empirical cumulative distribution function of improvement, which shows the percentage of test triples for which the level of improvement is less than or equal to a given value on each dataset. Surprisingly, the improvement for both MRR and Hits@10 is less than five percentage points for nearly 40% of FB15K-237's test set, and is *zero or negative* 15% of the time. By contrast, our baseline is significantly weaker than the strongest embedding method on CODEX-M.

The disparity in improvement is due to two relation patterns prevalent in FB15K-237:

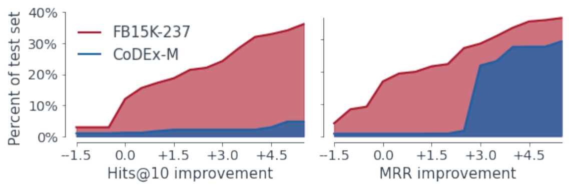

Figure 4: Empirical CDF of improvement of the best embedding over our frequency baseline.

- **Skewed relations** FB15K-237 contains many relations that are skewed toward a single head or tail entity. For example, our baseline achieves perfect performance over all $(h, r, ?)$ queries for the */common/topic/webpage. /common/webpage/category* relation because this relation has only *one* unique tail entity. Another example of a highly skewed relation in FB15K-237 is */people/person/gender*, for which 78.41% of tails are the entity *male*. In fact, 11 relations in FB15K-237 have only one unique tail entity, accounting for 3.22% of all tail queries in FB15K-237. Overall, 15.98% of test triples in FB15K-237 contain relations that are skewed 50% or more toward a single head or tail entity, whereas only 1.26% of test triples in CoDEx-M contain such skewed relations.

- **Fixed-set relations** Around 12.7% of test queries in FB15K-237 contain relation types that connect entities to *fixed sets of values*. As an example, each head entity that participates in the FB15K-237 relation */travel/travel_destination/climate./travel/travel_destination_monthly_climate/month* is connected to the *same* 12 tails (months of the year) throughout train, validation, and test. This makes prediction trivial with our baseline: By filtering out the tail entities already seen in train, only a few (or even one) candidate tail(s) are left in test, and the answer is guaranteed to be within these candidates. These relations only occur in FB15K-237 because of the way the dataset was constructed from Freebase. Specifically, Freebase used a special type of entity called Compound Value Type (CVT) as an intermediary node connecting *n-ary* relations. Binary relations were created by traversing through CVTs, yielding some relations that connect entities to fixed sets of values.

We conclude that while FB15K-237 is a valuable dataset, CoDEx is more appropriately difficult for link prediction. Additionally, we note that in FB15K-237, all validation and test triples containing entity pairs directly linked in the training set were deleted (Toutanova and Chen, 2015), meaning that symmetry cannot be tested for in FB15K-237. Given that CoDEx datasets contain both symmetry and compositionality, CoDEx is more suitable for assessing how well models can learn relation patterns that go beyond frequency.

7 Conclusion and outlook

We present **CoDEx**, a set of knowledge graph **Co**mpletion **D**atasets **Ex**tracted from Wikidata and Wikipedia, and show that CoDEx is suitable for multiple KGC tasks. We release data, code, and pretrained models for use by the community at https://bit.ly/2EPbrJs. Some promising future directions on CoDEx include:

- **Better model understanding** CoDEx can be used to analyze the impact of hyperparameters, training strategies, and model architectures in KGC tasks.

- **Revival of triple classification** We encourage the use of triple classification on CoDEx in addition to link prediction because it directly tests discriminative power.

- **Fusing text and structure** Including text in both the link prediction and triple classification tasks should substantially improve performance (Toutanova et al., 2015). Furthermore, text can be used for few-shot link prediction, an emerging research direction (Xiong et al., 2017; Shi and Weninger, 2017).

Overall, we hope that CoDEx will provide a boost to research in KGC, which will in turn impact many other fields of artificial intelligence.

Acknowledgments

The authors thank Michał Rybak and Xinyi (Carol) Zheng for their contributions. This material is supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, and an NSF Graduate Research Fellowship.

References

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *SIGMOD*.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019a. Multi-relational poincaré graph embeddings. In *NeurIPS*.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019b. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2n: Attending to neighbors for knowledge graph inference. In *ACL*.

Vladimir Batagelj and Matjaž Zaveršnik. 2011. Fast algorithms for determining (generalized) core groups in social networks. *ADAC*, 5(2).

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Guillaume Bouchard, Sameer Singh, and Theo Trouillon. 2015. On approximate reasoning capabilities of low-rank vector spaces. In *AAAI Spring Symposium Series*.

Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *NAACL-HLT*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *AAAI*.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*.

Alberto Garcia-Duran, Antoine Bordes, and Nicolas Usunier. 2015. Composing relationships with translations. In *EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Leo A Goodman. 1961. Snowball sampling. *Ann. Math. Stat.*

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *ACL-IJCNLP*.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *AAAI*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *CogSci*.

Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. 2020. Knowledge base completion: Baseline strikes back (again). *arXiv preprint arXiv:2005.00804*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL-IJCNLP*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*.

Yantao Jia, Yuanzhuo Wang, Hailun Lin, Xiaolong Jin, and Xueqi Cheng. 2016. Locally adaptive translation for knowledge graph embedding. In *AAAI*.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *NAACL-HLT*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *ACL RepL4NLP Workshop*.

Lucie-Aimée Kaffee, Alessandro Piscopo, Pavlos Vougiouklis, Elena Simperl, Leslie Carr, and Lydia Pintscher. 2017. A glimpse into babel: an analysis of multilinguality in wikidata. In *OpenSym*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*.

Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. 2006. Learning systems of concepts with an infinite relational model. In *AAAI*.

Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *ICML*.

Jonathan Lajus, Luis Galárraga, and Fabian M. Suchanek. 2020. Fast and exact rule mining with amie 3. In *ESWC*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation learning with entities, attributes and relations. In *IJCAI*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *ICML*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Alexa T McCray. 2003. An upper-level ontology for the biomedical domain. *Comp. Funct. Genomics*, 4(1).

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *ISWC*.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhanava Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *CACM*, 61(5):103–115.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *ACL*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *NAACL-HLT*.

Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1).

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2020. Revisiting evaluation of knowledge base completion models. In *AKBC*.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. In *ICLR*.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *WWW*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *ACL CVSC Workshop*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Théo Trouillon, Éric Gaussier, Christopher R Dance, and Guillaume Bouchard. 2019. On inductive abilities of latent factor models for relational learning. *JAIR*, 64.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020a. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *AAAI*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020b. Composition-based multi-relational graph convolutional networks. In *ICLR*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *CACM*, 57(10).

Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *NAACL-HLT*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12).

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *IJCAI*.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *IJCAI*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016a. From one point to a manifold: knowledge graph embedding for precise link prediction. In *IJCAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016b. Transg: A generative model for knowledge graph embedding. In *ACL*.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*.

Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. In *ACL*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. 2020. Relational graph neural network with hierarchical attention for knowledge graph completion. In *AAAI*.

A Literature review

Table 8 provides an overview of knowledge graph embedding papers with respect to datasets and evaluation tasks. In our review, we only consider papers published between 2014 and 2020 in the main proceedings of conferences where KGC embedding papers are most likely to appear: Artificial intelligence (AAAI, IJCAI), machine learning (ICML, ICLR, NeurIPS), and natural language processing (ACL, EMNLP, NAACL).

The main evaluation benchmarks are **FB15K** (Bordes et al., 2013), **WN18** (Bordes et al., 2013), **FB15K-237** (Toutanova and Chen, 2015), **WN18RR** (Dettmers et al., 2018), **FB13** (Socher et al., 2013), **WN11** (Socher et al., 2013), **NELL-995** (Xiong et al., 2017), **YAGO3-10** (Dettmers et al., 2018), **Countries** (Bouchard et al., 2015). **UMLS** (McCray, 2003), **Kinship** (Kemp et al., 2006), **Families** (Hinton, 1986), and other versions of **NELL** (Mitchell et al., 2018).

B Seeds for data collection

Table 9 provides all seed entity and relation types used to collect CoDEx. Each type is given first by its natural language label and then by its Wikidata unique ID: Entity IDs begin with Q, whereas relation (property) IDs begin with P. For the entity types that apply to *people* (e.g., actor, musician, journalist), we retrieved seed entities by querying Wikidata using the *occupation* relation. For the entity types that apply to *things* (e.g., airline, disease, tourist attraction), we retrieved seed entities by querying Wikidata using the *instance of* and *subclass of* relations.

C Negative annotation guidelines

We provide the annotation guidelines we used to label candidate negative triples (§ 3.4).

Task You must label each triple as either *true* or *false*. To help you find the answer, we have provided you with Wikipedia and Wikidata links for the entities and relations in each triple. You may also search on Google for the answer, although most claims should be resolvable using Wikipedia and Wikidata alone. If you are not able to find any reliable, specific, clear information supporting the claim, choose *false*. You may explain your reasoning if need be or provide sources to back up your answer in the optional explanation column.

Examples False triples may have problems with grammar, factual content, or both. Examples of *grammatically incorrect* triples are those whose entity or relation types do not make sense, for example:

- (*United States of America*, *continent*, *science fiction writer*)
- (*Mohandas Karamchand Gandhi*, *medical condition*, *British Raj*)
- (*Canada*, *foundational text*, *Vietnamese cuisine*)

Examples of grammatically correct but *factually false* triples include:

- (*United States of America*, *continent*, *Europe*)
- (*Mohandas Karamchand Gandhi*, *country of citizenship*, *Argentina*)
- (*Canada*, *foundational text*, *Harry Potter and the Goblet of Fire*)
- (*Alexander Pushkin*, *influenced by*, *Leo Tolstoy*) — Pushkin died only a few years after Tolstoy was born, so this sentence is unlikely.

Notice that in the latter examples, the entity types match up, but the statements are still false.

Tips For triples about people's *occupation* and *genre*, try to be as specific as possible. For example, if the triple says (<person>, *occupation*, *guitarist*) but that person is mainly known for their singing, choose *false*, even if that person plays the guitar. Likewise, if a triple says (<person>, *genre*, *classical*) but they are mostly known for jazz music, choose *false* even if, for example, that person had classical training in their childhood.

D Embedding models

We briefly overview the five models compared in our link prediction and triple classification tasks.

RESCAL (Nickel et al., 2011) was one of the first knowledge graph embedding models. Although it is not often used as a baseline, Ruffinelli et al. (2020) showed that it is competitive when appropriately tuned. RESCAL treats relational learning as tensor decomposition, scoring entity embeddings $\mathbf{h}, \mathbf{r} \in \mathbb{R}^{d_e}$ and relation embeddings $\mathbf{R} \in \mathbb{R}^{d_e \times d_e}$ with the bilinear form $\mathbf{h}^\top \mathbf{R} \mathbf{t}$.

Table 8: An overview of knowledge graph embedding papers published between 2014 and 2020 with respect to datasets and evaluation tasks. Original citations for datasets are given in Appendix A. Link pred. refers to link prediction, and triple class. refers to triple classification, both of which are covered in § 5.

		Datasets							Evaluation tasks		
	Reference	FB15K	FB15K-237	FB13	WN18	WN18RR	WN11	Other	Link pred.	Triple class.	Other
AAAI, IJCAI	(Wang et al., 2014)	✓		✓	✓		✓	FB5M	✓	✓	relation extraction (FB5M)
	(Lin et al., 2015b)	✓		✓	✓		✓	FB40K	✓	✓	relation extraction (FB40K)
	(Wang et al., 2015)							NELL (Location, Sports)	✓		
	(Nickel et al., 2016)	✓			✓			Countries	✓		
	(Lin et al., 2016)							FB24K	✓		
	(Wang and Cohen, 2016)	✓			✓				✓		
	(Xiao et al., 2016a)	✓		✓	✓		✓		✓	✓	
	(Jia et al., 2016)	✓		✓	✓		✓		✓	✓	
	(Xie et al., 2016)	✓						FB15K+	✓	✓	
	(Shi and Weninger, 2017)	✓						SemMedDB, DBPedia	✓		fact checking (not on FB15K)
	(Dettmers et al., 2018)	✓	✓		✓	✓		YAGO3-10, Countries	✓		
	(Ebisu and Ichise, 2018)	✓			✓				✓		
	(Guo et al., 2018)	✓						YAGO37	✓		
	(Zhang et al., 2020)	✓	✓		✓	✓			✓		
	(Vashishth et al., 2020a)		✓			✓		YAGO3-10	✓		
ICML, ICLR, NeurIPS	(Yang et al., 2015)	✓			✓			FB15K-401	✓		rule extraction (FB15K-401)
	(Trouillon et al., 2016)	✓			✓				✓		
	(Liu et al., 2017)	✓			✓				✓		
	(Kazemi and Poole, 2018)	✓			✓				✓		
	(Das et al., 2018)		✓			✓		NELL-995, UMLS, Kinship, Countries, WikiMovies	✓		QA (WikiMovies)
	(Lacroix et al., 2018)	✓	✓		✓	✓		YAGO3-10	✓		
	(Guo et al., 2019)	✓	✓		✓			DBPedia-YAGO3, DBPedia-Wikidata	✓		entity alignment (DBPedia graphs)
	(Sun et al., 2019)	✓	✓		✓	✓			✓		
	(Zhang et al., 2019)	✓	✓		✓	✓			✓		
	(Balazevic et al., 2019a)		✓			✓			✓		
	(Vashishth et al., 2020b)		✓			✓		MUTAG, AM, PTC	✓		graph classification (MUTAG, AM, PTC)
ACL, EMNLP, NAACL	(Ji et al., 2015)	✓		✓	✓		✓		✓	✓	
	(Guo et al., 2015)							NELL (Location, Sports, Freq)	✓	✓	
	(Guu et al., 2015)			✓			✓		✓	✓	
	(Garcia-Duran et al., 2015)	✓						Families	✓		
	(Lin et al., 2015a)	✓						FB40K	✓		relation extraction (FB40K)
	(Xiao et al., 2016b)	✓		✓	✓		✓		✓	✓	
	(Nguyen et al., 2016)	✓			✓				✓		
	(Xiong et al., 2017)		✓					NELL-995	✓		rule mining
	(Lin et al., 2018)		✓			✓		NELL-995, UMLS, Kinship	✓		
	(Nguyen et al., 2018)		✓			✓			✓		
	(Bansal et al., 2019)		✓			✓			✓		
	(Xu and Li, 2019)	✓	✓		✓	✓		YAGO3-10, Family	✓		
	(Balazevic et al., 2019b)	✓	✓		✓	✓			✓		
	(Vu et al., 2019)		✓			✓		SEARCH17	✓		personalized search (SEARCH17)
	(Nathani et al., 2019)		✓			✓		NELL-995, UMLS, Kinship	✓		
	(Jiang et al., 2019)	✓	✓		✓	✓			✓		

Table 9: The entity and relation types (Wikidata IDs in parentheses) used to seed CoDEx.

	Seed types
Entities	actor (Q33999), airline (Q46970), airport (Q1248784), athlete (Q2066131), book (Q571), businessperson (Q43845), city (Q515), company (Q783794), country (Q6256), disease (Q12136), engineer (Q81096), film (Q11424), government agency (Q327333), journalist (Q1930187), lake (Q23397), monarch (Q116), mountain (Q8502), musical group (Q215380), musician (Q639669), newspaper (Q11032), ocean (Q9430), politician (Q82955), record label (Q18127), religion (Q9174), religious leader (Q15995642), religious text (Q179461), scientist (Q901), sports league (Q623109), sports team (Q12973014), stadium (Q483110), television program (Q15416), tourist attraction (Q570116), visual artist (Q3391743), visual artwork (Q4502142), writer (Q36180)
Relations	airline alliance (P114), airline hub (P113), architect (P84), architectural style (P149), author (P50), capital (P36), cast member (P161), cause of death (P509), chairperson (P488), chief executive officer (P169), child (P40), continent (P30), country (P17), country of citizenship (P27), country of origin (P495), creator (P170), diplomatic relation (P530), director (P57), drug used for treatment (P2176), educated at (P69), employer (P108), ethnic group (P172), field of work (P101), foundational text (P457), founded by (P112), genre (P136), head of government (P6), head of state (P35), headquarters location (P159), health specialty (P1995), indigenous to (P2341), industry (P452), influenced by (P737), instance of (P31), instrument (P1303), language of work or name (P407), languages spoken, written, or signed (P1412), legal form (P1454), legislative body (P194), located in the administrative terroritorial entity (P131), location of formation (P740), medical condition (P1050), medical examinations (P923), member of (P463), member of political party (P102), member of sports team (P54), mountain range (P4552), movement (P135), named after (P138), narrative location (P840), notable works (P800), occupant (P466), occupation (P106), official language (P37), parent organization (P749), part of (P361), place of birth (P19), place of burial (P119), place of death (P20), practiced by (P3095), product or material produced (P1056), publisher (P123), record label (P264), regulated by (P3719), religion (P140), residence (P551), shares border with (P47), sibling (P3373), sport (P641), spouse (P26), studies (P2578), subclass of (P279), symptoms (P780), time period (P2348), tributary (P974), unmarried partner (P451), use (P366), uses (P2283)

TransE (Bordes et al., 2013) treats relations as translations between entities, i.e., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ for $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^{d_e}$, and scores embeddings with negative Euclidean distance $-\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$. TransE is likely the most popular baseline for KGC tasks and the most influential of all KGC embedding papers.

ComplEx (Trouillon et al., 2016) uses a bilinear function to score triples with a diagonal relation embedding matrix and complex-valued embeddings. Its scoring function is $\text{re}\left(\mathbf{h}^\top \text{diag}(\mathbf{r})\overline{\mathbf{t}}\right)$, where $\overline{\mathbf{t}}$ is the complex conjugate of $\mathbf{t}$ and re denotes the real part of a complex number.

ConvE (Dettmers et al., 2018) is one of the first and most popular *nonlinear* models for KGC. It concatenates head and relation embeddings $\mathbf{h}$ and $\mathbf{r}$ into a two-dimensional "image", applies a pointwise linearity over convolutional and fully-connected layers, and multiplies the result with the tail embedding $\mathbf{t}$ to obtain a score. Formally, its scoring function is given as $f(\text{vec}(f([\overline{\mathbf{h}}; \overline{\mathbf{r}}] * \omega))\mathbf{W})\mathbf{t}$, where f is a nonlinearity (originally, ReLU), $[\overline{\mathbf{h}}; \overline{\mathbf{r}}]$ denotes a concatenation and two-dimensional reshaping of the head and relation embeddings, ω denotes the filters of the convolutional layer, and vec denotes the flattening of a two-dimensional matrix.

TuckER (Balazevic et al., 2019b) is a linear model based on the Tucker tensor decomposition, which factorizes a tensor into three lower-rank matrices and a core tensor. The TuckER scoring function for a single triple (h, r, t) is given as $\mathcal{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t}$, where $\mathcal{W}$ is the mode-three core tensor that is shared among all entity and relation embeddings, and $\times_n$ denotes the tensor product along the n-th mode of the tensor. TuckER can be seen as a generalized form of other linear KGC embedding models like RESCAL and ComplEx.

E Content comparison

We provide additional comparison of the contents in CoDEx-M and FB15K-237.

Figure 5, which plots the **top-30 entities by frequency** in the two benchmarks, demonstrates that both dataset are biased toward developed Western countries and cultures. However, CoDEx-M is more diverse in domain. It covers academia, entertainment, journalism, politics, science, and writing, whereas FB15K-237 covers mostly entertaiment and sports. FB15K-237 is also much more biased toward the United States in particular, as five of its top-30 entities are specific to the US: *United States of America*, *United States dollar*, *New York City*,

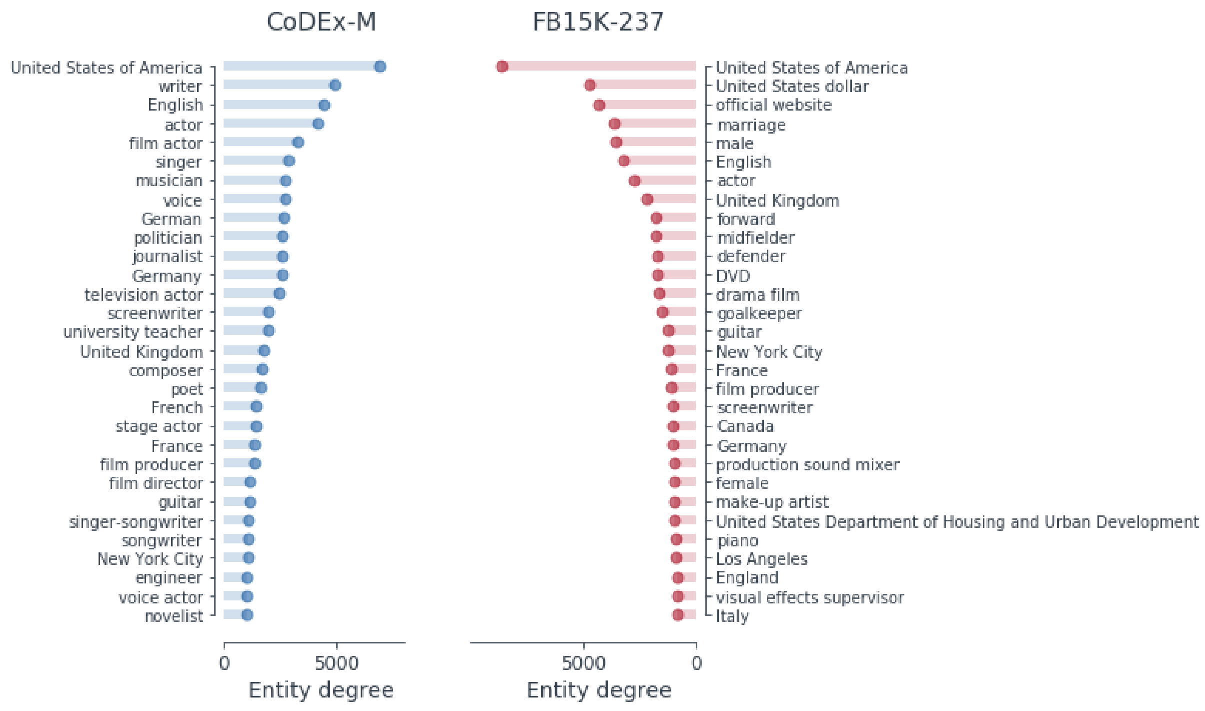

Figure 5: Top-30 **most frequent entities** in CoDEx-M and FB15K-237.

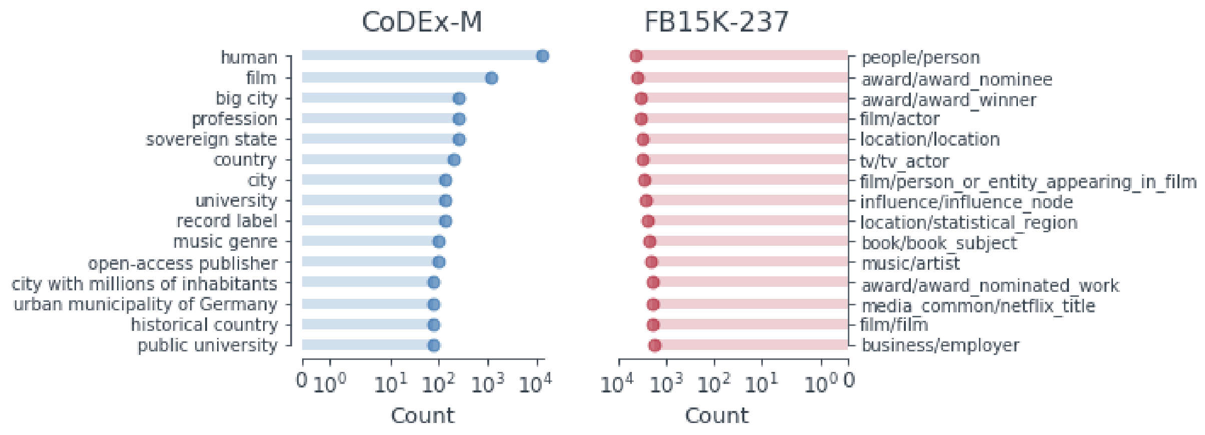

Figure 6: Top-15 **most frequent entity types** in CoDEx-M and FB15K-237.

Los Angeles, and the *United States Department of Housing and Urban Development*.

Figure 6 compares the **top-15 entity types** in CoDEx-M and FB15K-237. Again, CoDEx-M is diverse, covering people, places, organizations, movies, and abstract concepts, whereas FB15K-237 has many overlapping entity types mostly about entertainment.

F Hyperparameter search

Table 10 gives our hyperparameter search space. Tables 11, 12, and 13 report the best hyperparameter configurations for link prediction on CoDEx-S, CoDEx-M, and CoDEx-L, respectively. Tables 14 and 15 report the best hyperparameter configurations for triple classification on the hard negatives in CoDEx-S and CoDEx-M, respectively.

Terminology For embedding initialization, **Xv** refers to Xavier initialization (Glorot and Bengio, 2010). The **reciprocal relations** model refers to learning separate relation embeddings for queries in the direction of $(h, r, ?)$ versus $(?, r, t)$ (Kazemi and Poole, 2018). The **frequency weighting** regularization technique refers to regularizing embeddings by the relative frequency of the corresponding entity or relation in the training data.

Search strategies Recall that we select models using Ax, which supports hyperparameter search using both quasi-random sequences of generated configurations and Bayesian optimization (BO). The search strategy for each CoDEx dataset is as follows:

- **CoDEx-S**: Per negative sampling type/loss combination, we generate 30 quasi-random trials followed by 10 BO trials. We select the best-performing model by validation MRR over all such combinations. In each trial, the model is trained for a maximum of 400 epochs with an early stopping patience of 5. We also terminate a trial after 50 epochs if the model does not reach ≥ 0.05 MRR.
- **CoDEx-M**: Per negative sampling type/loss combination, we generate 20 quasi-random trials. The maximum number of epochs and early stopping criteria are the same as for CoDEx-S.
- **CoDEx-L**: Per negative sampling type/loss combination, we generate 10 quasi-random trials of 20 training epochs instead of 400. We reduce the number of epochs to limit resource usage. In most cases, MRR plateaus after 20-30 epochs, an observation which is consistent with (Ruffinelli et al., 2020). Then, we take the best-performing model by validation MRR over all such combinations, and retrain that model for a maximum of 400 epochs.

Note that we search using MRR as our metric, but the triple classification task measures 0/1 accuracy, not ranking performance. For triple classification, we choose the model with the highest validation accuracy among the pre-trained models across all negative sampling type/loss function combinations.

We release all pretrained LibKGE models and accompanying configuration files in the centralized CoDEx repository.

Table 10: Our hyperparameter search space. We follow the naming conventions and ranges given by Ruffinelli et al. (2020), and explain the meanings of selected hyperparameter settings in Appendix F. As most KGC embedding models have a wide range of configuration options, we encourage future work to follow this tabular scheme for transparent reporting of implementation details.

Hyperparameter	Range
Embedding size	$\{128, 256, 512\}$
Training type	{NegSamp, 1vsAll, KvsAll}
Reciprocal	{True, False}
# head samples (NegSamp)	$[1, 1000]$, log scale
# tail samples (NegSamp)	$[1, 1000]$, log scale
Label smoothing (KvsAll)	$[0, 0.3]$
Loss	{MR, BCE, CE}
Margin (MR)	$[0, 10]$
ℓ_p norm (TransE)	$\{1, 2\}$
Optimizer	{Adam, Adagrad}
Batch size	$\{128, 256, 512, 1024\}$
Learning rate	$[10^{-4}, 1]$, log scale
LR scheduler patience	$[0, 10]$
ℓ_p regularization	$\{1, 2, 3,$ None$\}$
Entity embedding weight	$[10^{20}, 10^{-5}]$
Relation embedding weight	$[10^{20}, 10^{-5}]$
Frequency weighting	{True, False}
Embedding normalization (TransE)	
Entity	{True, False}
Relation	{True, False}
Dropout	
Entity embedding	$[0.0, 0.5]$
Relation embedding	$[0.0, 0.5]$
Feature map (ConvE)	$[0.0, 0.5]$
Projection (ConvE)	$[0.0, 0.5]$
Embedding initialization	{Normal, Unif, XvNorm, XvUnif}
Stdev (Normal)	$[10^{-5}, 1.0]$
Interval (Unif)	$[-1.0, 1.0]$
Gain (XvNorm)	1.0
Gain (XvUnif)	1.0

Table 11: Best **link prediction** hyperparameter configurations on **CODEX-S**.

	RESCAL	TransE	ComplEx	ConvE	TuckER
Best validation MRR	0.4076	0.3602	0.4752	0.4639	0.4574
Embedding size	512	512	512	256	512
Training type	1vsAll	NegSamp	1vsAll	1vsAll	KvsAll
Reciprocal	No	Yes	Yes	Yes	Yes
# head samples (NegSamp)	-	2	-	-	-
# tail samples (NegSamp)	-	56	-	-	-
Label smoothing (KvsAll)	-	-	-	-	0.0950
Loss	CE	CE	CE	CE	CE
Margin (MR)	-	-	-	-	-
ℓ_p norm (TransE)	-	2	-	-	-
Optimizer	Adagrad	Adagrad	Adam	Adagrad	Adagrad
Batch size	128	128	1024	512	256
Learning rate	0.0452	0.0412	0.0003	0.0117	0.0145
LR scheduler patience	7	6	7	3	1
ℓ_p regularization	3	2	None	3	1
Entity embedding weight	2.18×10^{-10}	1.32×10^{-7}	9.58×10^{-13}	3.11×10^{-15}	3.47×10^{-15}
Relation embedding weight	3.37×10^{-14}	3.72×10^{-18}	0.0229	4.68×10^{-9}	3.43×10^{-14}
Frequency weighting	False	False	True	True	True
Embedding normalization (TransE)					
Entity	-	No	-	-	-
Relation	-	No	-	-	-
Dropout					
Entity embedding	0.0	0.0	0.0793	0.0	0.1895
Relation embedding	0.0804	0.0	0.0564	0.0	0.0
Feature map (ConvE)	-	-	-	0.2062	-
Projection (ConvE)	-	-	-	0.1709	-
Embedding initialization	Normal	XvNorm	XvNorm	XvNorm	XvNorm
Stdev (Normal)	0.0622	-	-	-	-
Interval (Unif)	-	-	-	-	-
Gain (XvNorm)	-	1.0	1.0	1.0	1.0
Gain (XvUnif)	-	-	-	-	-

Table 12: Best **link prediction** hyperparameter configurations on **CoDEx-M**.

	RESCAL	TransE	ComplEx	ConvE	TuckER
Best validation MRR	0.3173	0.2993	0.3351	0.3146	0.3253
Embedding size	256	512	512	512	512
Training type	1vsAll	NegSamp	KvsAll	NegSamp	KvsAll
Reciprocal	Yes	Yes	Yes	Yes	Yes
# head samples (NegSamp)	-	2	-	381	-
# tail samples (NegSamp)	-	56	-	751	-
Label smoothing (KvsAll)	-	-	0.2081	-	0.0950
Loss	CE	CE	CE	CE	CE
Margin (MR)	-	-	-	-	-
ℓ_p norm (TransE)	-	2	-	-	-
Optimizer	Adagrad	Adagrad	Adagrad	Adagrad	Adagrad
Batch size	256	128	1024	128	256
Learning rate	0.0695	0.0412	0.2557	0.0024	0.0145
LR scheduler patience	8	6	6	9	1
ℓ_p regularization	2	2	3	1	1
Entity embedding weight	9.56×10^{-7}	1.32×10^{-7}	1.34×10^{-10}	1.37×10^{-10}	3.47×10^{-15}
Relation embedding weight	2.56×10^{-17}	3.72×10^{-18}	6.38×10^{-16}	$4.72 \times 10{-}10$	3.4×10^{-14}
Frequency weighting	False	False	True	True	True
Embedding normalization (TransE)					
Entity	-	No	-	-	-
Relation	-	No	-	-	-
Dropout					
Entity embedding	0.0	0.0	0.1196	0.0	0.1895
Relation embedding	0.0	0.0	0.3602	0.0348	0.0
Feature map (ConvE)	-	-	-	0.3042	-
Projection (ConvE)	-	-	-	0.2343	-
Embedding initialization	XvUnif	XvUnif	Unif	XvNorm	XvNorm
Stdev (Normal)	-	-	-	-	-
Interval (Unif)	-	-	-0.8133	-	-
Gain (XvNorm)	-	-	-	1.0	1.0
Gain (XvUnif)	1.0	1.0	-	-	-

Table 13: Best **link prediction** hyperparameter configurations on **CoDEx-L**.

	RESCAL	TransE	ComplEx	ConvE	TuckER
Best validation MRR	0.3030	0.1871	0.2943	0.3010	0.3091
Embedding size	128	128	128	256	256
Training type	1vsAll	NegSamp	1vsAll	1vsAll	1vsAll
Reciprocal	No	Yes	Yes	Yes	No
# head samples (NegSamp)	-	209	-	-	-
# tail samples (NegSamp)	-	2	-	-	-
Label smoothing (KvsAll)	-	-	-	-	-
Loss	CE	CE	CE	CE	CE
Margin (MR)	-	-	-	-	-
ℓ_p norm (TransE)	-	2	-	-	-
Optimizer	Adagrad	Adam	Adagrad	Adagrad	Adagrad
Batch size	1024	128	1024	256	512
Learning rate	0.2651	0.0009	0.2651	0.0329	0.0196
LR scheduler patience	7	9	7	1	4
ℓ_p regularization	2	2	2	1	2
Entity embedding weight	2.01×10^{-16}	7.98×10^{-14}	2.01×10^{-16}	6.10×10^{-16}	8.06×10^{-11}
Relation embedding weight	3.52×10^{-13}	3.42×10^{-9}	3.52×10^{-13}	1.03×10^{-16}	7.19×10^{-19}
Frequency weighting	True	False	True	True	True
Embedding normalization (TransE)					
Entity	-	No	-	-	-
Relation	-	No	-	-	-
Dropout					
Entity embedding	0.0	0.0	0.0	0.0064	0.1606
Relation embedding	0.0	0.0	0.0	0.0	0.0857
Feature map (ConvE)	-	-	-	0.1530	-
Projection (ConvE)	-	-	-	0.4192	-
Embedding initialization	Normal	Unif	Normal	XvNorm	Normal
Stdev (Normal)	0.0169	-	0.0169	-	0.0002
Interval (Unif)	-	-0.4464		-	-
Gain (XvNorm)	-	-		1.0	-
Gain (XvUnif)	-	-		-	-

Table 14: Best **triple classification** hyperparameter configurations on **CODEX-S** (**hard negatives**).

	RESCAL	TransE	ComplEx	ConvE	TuckER
Best validation accuracy	0.8571	0.8511	0.8558	0.8607	0.8596
Embedding size	See Tab. 11	See Tab. 11	See Tab. 11	512	See Tab. 11
Training type	1vsAll	NegSamp	1vsAll	1vsAll	KvsAll
Reciprocal	See Tab. 11	See Tab. 11	See Tab. 11	Yes	See Tab. 11
# head samples (NegSamp)	-	See Tab. 11	-	-	-
# tail samples (NegSamp)	-	See Tab. 11	-	-	-
Label smoothing (KvsAll)	-	-	-	-	-
Loss	CE	CE	CE	BCE	CE
Margin (MR)	-	-	-	-	-
ℓ_p norm (TransE)	-	See Tab. 11	-	-	-
Optimizer	See Tab. 11	See Tab. 11	See Tab. 11	Adagrad	See Tab. 11
Batch size	See Tab. 11	See Tab. 11	See Tab. 11	256	See Tab. 11
Learning rate	See Tab. 11	See Tab. 11	See Tab. 11	0.0263	See Tab. 11
LR scheduler patience	See Tab. 11	See Tab. 11	See Tab. 11	7	See Tab. 11
ℓ_p regularization	See Tab. 11	See Tab. 11	See Tab. 11	2	See Tab. 11
Entity embedding weight	See Tab. 11	See Tab. 11	See Tab. 11	9.62×10^{-6}	See Tab. 11
Relation embedding weight	See Tab. 11	See Tab. 11	See Tab. 11	1.34×10^{-12}	See Tab. 11
Frequency weighting	See Tab. 11	See Tab. 11	See Tab. 11	False	See Tab. 11
Embedding normalization (TransE)					
Entity	-	See Tab. 11	-	-	-
Relation	-	See Tab. 11	-	-	-
Dropout					
Entity embedding	See Tab. 11	See Tab. 11	See Tab. 11	0.1620	See Tab. 11
Relation embedding	See Tab. 11	See Tab. 11	See Tab. 11	0.0031	See Tab. 11
Feature map (ConvE)	-	-	-	0.0682	-
Projection (ConvE)	-	-	-	0.2375	-
Embedding initialization	See Tab. 11	See Tab. 11	See Tab. 11	Normal	See Tab. 11
Stdev (Normal)	See Tab. 11	See Tab. 11	See Tab. 11	0.0006	See Tab. 11
Interval (Unif)	See Tab. 11	See Tab. 11	See Tab. 11	-	See Tab. 11
Gain (XvNorm)	See Tab. 11	See Tab. 11	See Tab. 11	-	See Tab. 11
Gain (XvUnif)	See Tab. 11	See Tab. 11	See Tab. 11	-	See Tab. 11

Table 15: Best **triple classification** hyperparameter configurations on **CoDEx-M** (**hard negatives**).

	RESCAL	TransE	ComplEx	ConvE	TuckER
Best validation accuracy	0.8232	0.8002	0.8267	0.8292	0.8267
Embedding size	512	See Tab. 12	512	512	See Tab. 12
Training type	KvsAll	NegSamp	KvsAll	KvsAll	KvsAll
Reciprocal	Yes	See Tab. 12	Yes	Yes	See Tab. 12
# head samples (NegSamp)	-	See Tab. 12	-	-	-
# tail samples (NegSamp)	-	See Tab. 12	-	-	-
Label smoothing (KvsAll)	0.0949	-	0.2081	0.0847	-
Loss	CE	CE	CE	CE	CE
Margin (MR)	-	-	-	-	-
ℓ_p norm (TransE)	-	See Tab. 12	-	-	-
Optimizer	Adagrad	See Tab. 12	Adagrad	Adagrad	See Tab. 12
Batch size	256	See Tab. 12	1024	1024	See Tab. 12
Learning rate	0.0144	See Tab. 12	0.2557	0.0378	See Tab. 12
LR scheduler patience	1	See Tab. 12	6	6	See Tab. 12
ℓ_p regularization	1	See Tab. 12	3	3	See Tab. 12
Entity embedding weight	3.47×10^{-15}	See Tab. 12	1.34×10^{-10}	1.03×10^{-16}	See Tab. 12
Relation embedding weight	3.43×10^{-14}	See Tab. 12	6.38×10^{-16}	0.0052	See Tab. 12
Frequency weighting	True	See Tab. 12	True	True	See Tab. 12
Embedding normalization (TransE)					
Entity	-	See Tab. 12	-	-	-
Relation	-	See Tab. 12	-	-	-
Dropout					
Entity embedding	0.1895	See Tab. 12	0.1196	0.4828	See Tab. 12
Relation embedding	0.0	See Tab. 12	0.3602	0.0	See Tab. 12
Feature map (ConvE)	-	-	-	0.2649	-
Projection (ConvE)	-	-	-	0.2790	-
Embedding initialization	XvNorm	See Tab. 12	Unif	XvUnif	See Tab. 12
Stdev (Normal)	-	See Tab. 12	-	-	See Tab. 12
Interval (Unif)	-	See Tab. 12	-0.8133	-	See Tab. 12
Gain (XvNorm)	1.0	See Tab. 12	-	-	See Tab. 12
Gain (XvUnif)	-	See Tab. 12	-	1.0	See Tab. 12