Quantifying the Impact of Predicate Similarities on Knowledge Graph Triple Embeddings

Alexander Kalinowski[1,*], Yuan An[1]

[1]Drexel University, Department of Information Science, 3141 Chestnut Street, Philadelphia, PA 19104

Abstract

In devising methods for representing knowledge graph *triples* in low-dimensional spaces, care must be taken to quantify the similarities between all components, especially the predicate components common to all triples. Unfortunately, knowledge graph benchmarks do not come equipped with scores indicating the semantic similarity between two arbitrary triples. To proxy these scores, we introduce a weakly supervised method we call PTSS, or pairwise triple similarity scoring. A neural model then utilizes this information to update triple representations. We conduct experiments using this method by substituting three methods for predicate relatedness measures: linear algebraic similarities of predicate embeddings, predicate frequency/inverse predicate frequency, and KL-divergence of predicate distributions. We analyze the information captured by these approaches and their impacts when utilized as weak supervision signals for triple representations to test the hypothesis that these approaches reflect a notion of cognitive similarity. Our findings indicate a combined model using scores driven from neural embeddings for entities and fact distributions for predicates achieves the best results, highlighting the efficacy of combining neural and distributional approaches and suggests this pattern of combination may be fruitful in other cognitively inspired AI solutions.

Keywords

knowledge representation, semantic web, triple embedding, knowledge graph embedding, predicate similarities

1. Introduction

Knowledge graphs (KGs) serve as a representation framework for real-world facts, stored as triples featuring a head entity h, a tail entity t and a predicate p expressing a relation between h and t. Knowledge graphs may store millions of such triples; determining how similar each triple is to all other triples in the graph is still an area in need of development. The majority of knowledge graph research is in link prediction and is heavily dependent on latent vector representations, herein referred to as knowledge graph embeddings (KGE). Treating the heads, tails and predicates as distinct embedding vectors requires the combination of all three pieces to arrive at a single vector representation for a triple. To this end, we propose a weakly supervised method called PTSS (pairwise triple similarity scoring) for triple embeddings. The method utilizes existing knowledge graph embeddings to create weakly supervised semantic

AIC 2022, 8th International Workshop on Artificial Intelligence and Cognition

*Corresponding author.

ajk437@drexel.edu (A. Kalinowski); ya45@drexel.edu (Y. An)

https://github.com/akalino (A. Kalinowski); https://github.com/anyuanay (Y. An)

similarity scores between triples. These scores can then be used for determining a single vector representation of a triple.

The PTSS method achieves significant improvements in two downstream applications, *triple classification* and *triple clustering* tasks, compared to the state-of-the-art baselines. The detailed design and evaluation of the proposed PTSS method is under review for publication. In this workshop paper, we expand on this work and probe the efficacy of three different scoring functions of predicate similarity. The rest of the paper begins with the problem formulation and study motivation, followed by details on the PTSS method. The remaining sections describe our results on evaluating the impacts of different predicate similarity functions using an *information theoretic metric* and a *downstream triple classification task*.

2. Problem Formulation and Motivation

We begin with definitions of knowledge graphs and knowledge graph embeddings, followed by motivation for why triple embedding methods require novel approaches. A knowledge graph (KG) $G = \{(h, p, t)\} \subset \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ is comprised of triples $T = (h, p, t)$, where $h \in \mathcal{E}$ is the head entity, $t \in \mathcal{E}$ is the tail entity and $p \in \mathcal{P}$ is a predicate, or relation, between h and t. Here, $\mathcal{E}$ and $\mathcal{P}$ are, respectively, the sets of entities and predicates; we denote the number of entities as $\#\mathcal{E}$ and number of predicates $\#\mathcal{P}$. A knowledge graph embedding (KGE) method encodes the patterns and regularities in a knowledge graph as low-dimensional, dense vectors, where entities $v \in E$ are d-dimensional vectors $\mathbf{e}_v \in \mathbb{R}^d$ and predicates $r \in P$ are scoring functions $f_r : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Each KGE method applies a different scoring function, and, as a result, predicates can be embedded as a vector $\mathbf{e}_r \in \mathbb{R}^k$ as in TransE [1] or a matrix $\mathbf{E}_r \in \mathbb{R}^{m \times n}$ as in DistMult [2]. A triple embedding (TE) method encodes the regularity of a triple $\langle h, p, t \rangle$ in a knowledge graph as a dense numeric vector. A highly applicable regularity among triples is the semantic similarity between triples. In particular, if two triples share one or more common elements, the embeddings of the triples should reflect a certain degree of geometrical proximity.

2.1. Motivation: Semantic Aggregation of KGEs for TEs

A great deal of effort has been put into developing KGE methods (see Section 7 for more details.) Let f be an arbitrary KGE model which generates a set of vectors $\mathbf{e}_v$ for entities and $\mathbf{e}_r$ for predicates. Let $k_1 = \langle h_1, p_1, t_1 \rangle$ be an arbitrary triple from the graph. To move into a space that is representative of the triple, we can exploit the existing KGE by aggregating the embedding vectors of its components for the whole triple embedding, i.e.,

$$\mathbf{e}_{k_1} = agg(\mathbf{e}_{h_1}, \mathbf{e}_{p_1}, \mathbf{e}_{t_1}).$$

As many of these models are built for link prediction, meaningful aggregations only make use of entity embeddings. Table 1 summarizes several common aggregation operations in the literature.

The issue with this type of aggregation comes when two triples share the same head and tail entities, but differing predicates. Let $k_1 = \langle h_1, p_1, t_1 \rangle$ and $k_2 = \langle h_1, p_2, t_1 \rangle$ be two such triples. Substituting into the aggregations in Table 1, we find that each representation would be exactly

Table 1
A summary of aggregation operators for constructing triple representations

Operator	Definition
Average (AVG)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = \frac{f_i(\mathbf{u})+f_i(\mathbf{v})}{2}$
Hadamard (HAD)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = f_i(\mathbf{u}) * f_i(\mathbf{v})$
Weighted-L1 (L1)	$\lvert f_i(\mathbf{u}) - f_i(\mathbf{v})\rvert$
Weighted-L2 (L2)	$\lvert f_i(\mathbf{u}) - f_i(\mathbf{v})\rvert^2$
Concatenation (HT)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = f_i(\mathbf{u}) \Vert f_i(\mathbf{v})$

Notes:
'$*$' denotes element-wise product operation.
'$\Vert$' denotes vector concatenation operation.

the same, although it is obvious that k_1 and k_2 are representing two different facts in the graph. This motivates research in finding methods to represent triples in a more semantic way that respects the information contained in the predicates.

3. PTSS: A Weakly Supervised Method for Building Triple Embeddings

Our main goal in this line of research is to build a method for embedding knowledge graph triples in a semantically oriented way, namely, respecting that triples sharing similar entities, predicates, or meanings should have high similarity when compared in their latent spaces. For this purpose, we propose a weakly supervised method called pairwise triple similarity scoring (PTSS).

Let G be a KG and f a corresponding KGE function. For triples $\langle h, p, t\rangle$, the embeddings learned through f may already encode vector space notions of similarities between entities and predicates. For example, two entities sharing many properties and relationships will participate in many of the same triples, and we assume that their respective embeddings should exhibit these similarities. This assumption is highly dependent on the selection of the pre-trained embedding function f, and there exists a line of research into how much semantic information these embeddings actually encode [3]. We can use these embeddings, henceforth referred to as *seed embeddings*, as a means to proxy pairwise triple similarity scores. Specifically, given two triples $T_a = \langle h_a, p_a, t_a\rangle$ and $T_b = \langle h_b, p_b, t_b\rangle$ in the knowledge graph, we define the *pairwise triple similarity score (PTSS)* between T_a and T_b as an average of the similarities between the embeddings of their head entities, tail entities, and predicates, respectively.

Formally, the PTSS is defined as

$$PTSS(T_a, T_b) = avg(sim(\mathbf{e}_{h_a}, \mathbf{e}_{h_b}), sim(\mathbf{e}_{p_a}, \mathbf{e}_{p_b}), sim(\mathbf{e}_{t_a}, \mathbf{e}_{t_b})),$$

where $\mathbf{e}_{h_a}$ is the embedding vector of the head entity h_a and so on, $avg(x, y, z)$ computes an average of its arguments, and the function $sim(\mathbf{e}_1, \mathbf{e}_2)$ is a function for computing the similarities of two embeddings. The similarity function $sim(\mathbf{e}_1, \mathbf{e}_2)$ could be an arbitrary function capturing the geometric structures in KG embedding spaces. The current PTSS employs

cosine similarity and arithmetic mean, leaving the evaluation of more general functions to future work. Our evaluation demonstrates that these simple functions generate useful weak supervision signals.

For weakly supervised training, we extract from the knowledge graph a set of training examples containing both *"positive"* and *"negative"* examples. For a triple $T_a = \langle h_a, p_a, t_a \rangle$, we define positive training examples as a set of potential *candidate matches*. In particular, triple T_a is composed of three elements, or three potential 'slots' to match on: the head, the tail and the predicate. For each 'slot' we select N other triples with the same head entity, N other triples with the same tail entity, N other triples with the same predicate. For negative examples, we select N other triples with no commonalities in any available slot. In instances where the set of candidate matches has cardinality less than N, we add the entire set to be scored. Thus, for each triple in the graph, we build a set of at most $4N$ triples to compare and contrast with. In all following experiments, we set $N = 5$, leaving an ablation of this sampling hyperparameter to future work.

These scores and training examples can then be utilized as an input to a fine-tuning process to generate low-dimensional representations of the composite triple. In particular, we define a Siamese-like neural network that takes two triples as input to an embedding layer and initializes values at an embedding layer with aggregations of the embeddings of the triple components. The network subsequently forward propagates the embedding layer through a series of encoding (ENC) layers. The final layer is a scoring (SCO) layer where the network computes the cosine similarity of the representations. This score is then compared to the estimated pairwise similarity scores (PTSS), with the errors back-propagated through the network. To update the triple embeddings, we make the embedding layer tunable and use the final values as the triple embeddings.

4. Measuring Predicate Similarity in PTSS

In computing $PTSS(T_a, T_b)$, we have several choices for building the entity and predicate signals. This paper focuses on the selection of predicate similarity measures, $sim(\mathbf{e}_{p_a}, \mathbf{e}_{p_b})$. Specifically, we hone in on three major approaches: those based on predicate vector similarities, those based on predicate frequency measures, and those based on fact distributions with respect to predicates. Each of these then defines a symmetric similarity matrix M_p used as input to the PTSS scoring function.

Predicate Vector Similarity. We utilize the similarity functions shown in Table 2 for similarities based on predicate vectors. The column 'Method' lists the embedding methods selected for this study (see Section 5.1). The pairwise similarity functions give rise to a symmetric similarity matrix M_p . We refer to this approach as **EMB**.

Predicate Frequency Measure. For similarities based on predicate frequency measures, we use the method of [4], who define triple frequency and inverse triple frequency as

$$TF(p_i, p_j) = \log(1 + \mathbf{C}_{i,j})$$

$$ITF(p_j, E) = \log \frac{|E|}{|p_i : \mathbf{C}_{i,j} > 0|}$$

Table 2
Knowledge graph embedding methods and functions for computing similarities of the resulting predicate embeddings.

Method	Similarity Function
ComplEx	$sim(p_1, p_2) = \exp(Re(p_1)^\top Re(p_2) / \left\|Re(p_1)\right\|_2 \left\|Re(p_2)\right\|_2)$
ConvE	$sim(p_1, p_2) = \exp(p_1^\top p_2 / \left\|p_1\right\|_2 \left\|p_2\right\|_2)$
DistMult	$sim(p_1, p_2) = \exp(p_1^\top p_2 / \left\|p_1\right\|_2 \left\|p_2\right\|_2)$
RESCAL	$sim(p_1, p_2) = \exp(\left\|M_{p_1} = M_{p_2}\right\|)$
RotatE	$sim(p_1, p_2) = \exp(-\Sigma_{i=1}^{n} \left\|p_{1,i} - p_{2,i}\right\|)$
TransE	$sim(p_1, p_2) = \exp(p_1^\top p_2 / \left\|p_1\right\|_2 \left\|p_2\right\|_2)$

where $C_{i,j}$ counts the number of times the predicates p_i and p_j link the same heads and tails, and E is the set of edges. These metrics are used to build a symmetric matrix

$$\mathbf{C}_M(i,j) = TF(p_i, p_j) \times ITF(p_j, E)$$

to represent a vector for each predicate in the graph; each vector in $\mathbf{C}_M$ can then be used to build the matrix M_p of pairwise similarities between predicates. Herein, we refer to this approach as **PF/IPF**.

Predicate Distribution Measure. An alternative quantification of predicate similarity can be found in [5], where conditional probabilities distributions based on the occurrences of h, p and t are used. Specifically, each predicate is assigned a conditional probability distribution $P(h, t|p)$, where similar predicates are expected to show similar probability distributions, i.e. if two predicates p_1 and p_2 share many of the same head, tail pairs, they should have similar probability mass for h, t. Here, rather than directly enumerate through all h, t pairs for each r, the conditional probability distribution is parameterized via a two latent neural network. This network can be expressed as

$$\tilde{u}_{\theta_1}(h; p) = \texttt{MLP}_{\theta_1}(\mathbf{p})^\top \mathbf{h},$$
$$\tilde{u}_{\theta_2}(t; h, p) = \texttt{MLP}_{\theta_2}([\mathbf{h}; \mathbf{p}])^\top \mathbf{t}$$

Here, $\mathbf{h}, \mathbf{p}, \mathbf{t}$ are embeddings of h, p, t, respectively. The output of this network can then be utilized to build a final conditional probability

$$P_{\theta^*}(h, t|p) = \exp(u_{\theta_1}(h; p) + u_{\theta_2}(t; h, p))$$

To compute the similarity between any arbitrary pair of predicates, the KL divergence of their respective conditional probability distributions is computed via a sampling-based approach. Important to this sampling approach is the number of samples K drawn from each conditional probability distribution when estimating the KL divergence; the authors fix this choice at 20, and we follow suit, leaving exploration of other choices to future work. For each experiment, we normalize and diagonalize the resulting KL divergence scores to build the matrix M_p of predicate similarities. For the remainder of this paper, we will refer to this method as **KL-div**.

Our goal is to compare and contrast these approaches, evaluating the information gained from each and their eventual impact on triple classification.

5. Evaluation Approach

With an interest in quantifying the amount of semantic information captured by both neural embedding models and information-theoretic approaches, we perform our experiments in two branches. First, we aim to quantify the amount of information contained in these representations, specifically in their pairwise similarities. Second, we aim to quantify the impact of this information on a task related to triple representations: triple classification. We outline these approaches in the following sections, followed by coverage of our benchmark datasets.

5.1. Selecting Representations

To evaluate the degree to which knowledge graph embeddings capture predicate semantics, we selected six popular neural embedding architectures and two information-theoretic models. For embeddings, we utilized pre-trained models from the [6] library, selecting **ComplEx**[7], **ConvE**[8], **DistMult**[2], **RotatE**[9], **RESCAL**[10], and **TransE**[1]. These models were selected for their variety in scoring functions and training hyperparameters, covering a large portion of research in this area while taking advantage of pre-trained model weights for initialisation. We cover the details of these approaches in Section 7.

5.2. Quantifying Predicate Representation Informativeness

To start, we wish to quantify the amount of information contained in the matrices M_p described above. One such numerical quantification is the von Neumann entropy of a matrix, defined as

$$\texttt{entropy} = -\operatorname{tr}(M_p) \cdot \log(M_p).$$

Here, we posit that higher entropy measures correspond to higher capture of information and should be an indicator of which representations are best. We further hypothesize that the representation of predicate similarities with the best entropy will maximize the impact on our downstream triple embedding tasks, detailed in Section 5.3.

In addition to numerical metrics on the information capture by each predicate similarity matrix, we are interested in the extent to which these similarities capture actual *semantics* about the graph. We perform an exploration of the semantics of each space, making sure that predicates that are ranked as 'most similar' are actually in-line with human expectations. We can also compare how similar each approach is by measuring the overlap in each of the similarity lists. To do so, for each pair of approaches, we compute the Jaccard similarity of the top five similar predicates for every predicate in the list. We then average over all predicates to arrive at a single metric. This approach allows us to quantify how similar the predicted predicate similarities are and determine which approaches have agreeable rankings. Approaches with highest overlapping predicate similarities are expected to have similar performance on the downstream tasks.

5.3. Probing Triple Embeddings

For each selected representation model (see Section 5.1), we apply the aggregation schemes found in Table 1 to build the initialization of the triple vectors. This creates five different PTSS

models for each selected representation scheme, for example, **TransE_AVG**, **TransE_HAD**, **TransE_L1**, **TransE_L2**, and **TransE_HT**. We then compute the PTSS scores using the **EMB**, the **PF/IPF** approach and the **KL-div** approach. This additionally creates three training datasets used as input to each model, for example,

TransE_AVG_EMB, **TransE_AVG_PFIPF**, **TransE_AVG_KLDIV**.

Thus, there are three models per aggregation, five aggregations per model and six representation approaches for a total of ninety models for each experiment. Each of these generates a unique triple representation used in the following evaluation tasks.

We probe the resulting triple embeddings for their ability to predict their respective predicate labels – a task of triple classification. Following the work of [11] and [12], we perform the classification using both a low-capacity and high-capacity model. For our low-capacity model, we employ one-vs-rest logistic regression, providing the model with triple representations as features and the predicate indices as labels. For the high-capacity model, we replace the logistic regression with a multi-layer perceptron with 512 hidden nodes, trained with the Adam optimizer and batch size of 256 for a total of 10 epochs. In each case, we split the triples into training and test sets, varying the number of training triples to be between 20% and 90% of all available triples, repeated using ten different random seeds. Both models are then evaluated on their respective Micro-F1 scores.

5.4. Knowledge Graph Benchmarks

In this work, we utilize two well-established knowledge graphs for evaluation, WordNet (WN18RR) and Freebase (FB15K-237). Summary statistics for both datasets are presented in Table 3. Freebase contains general facts about people, places and events. In our case, we selected the FB15K-237 benchmark, where inverse relations that are simple to learn (and thus inflate performance metrics) have been removed, leaving 237 distinct relations to be modelled. WordNet covers lexical and grammatical knowledge; we selected the WN18RR benchmark, which similarly removes inverse relations. These benchmark datasets were selected for two reasons. First, the frequency of usage in the research community allows us to take advantage of publicly available, pre-trained models, such as those found in LibKGE [6]. Secondly, these graphs are complex in nature, making our evaluation more reflective of knowledge graphs that occur in practice. Of major importance in our approach is determining the number of multi-edge triples, i.e. those that contain matching head and tail entities but multiple predicates. As outlined in Section 2, these are the predicates that suffer most from methods focused on aggregation of head and tail vectors for representing a triple. Our methodology is specifically focused on adding predicate information back to triple representations, capturing the semantics of these predicates and helping to build triple representations that can be differentiated along this dimension.

6. Analysis of Results

We present our findings in two categories: analysis of the information encoded in Section 6.1, and impact on downstream tasks in Section 6.2.

Table 3
Summary of Benchmarks

Dataset	WN18RR	FB15K-237
Entities	40,943	14,541
Predicates	11	237
Triples	93,003	310,116
Multi-edge Triples	218	49,214

6.1. Analysis of Available Information

In terms of entropy, the largest scores (and hence where most information is captured) come from the neural approaches for FB15K-237, but this is flipped for WN18RR, where information-theoretic approaches shine. We suspect this is largely due to the diversity in predicates; Freebase has many predicates and associated facts, allowing neural approaches more opportunities for learning while misleading information-theoretic approaches. WordNet, on the other hand, has fewer, more concentrated predicates and regularities in facts that information-theoretic approaches are likely to capture. From an entropy point-of-view, the best models of predicate similarity are **TransE** for FB15K-237 and **KL-div** for WN18RR, followed closely by **TransE**. This relationship is particularly interesting as **KL-div** is dependent on **TransE** embeddings as input, hence the close entropy scores should almost be expected, while the deviation in entropy for Freebase highlights an area for further investigation.

Table 4
von Neumann entropy measures for predicate representations.

Method	FB15k-237	WN18RR
KL-div	-6579.78	-97.42
PF/IPF	-7968.92	-61.62
ComplEx	-11802.18	-17.26
ConvE	-11334.13	-60.35
DistMult	-11711.66	-54.99
RESCAL	-2565.66	-52.88
RotatE	-10367.84	-55.15
TransE	-17233.54	-70.79

For FB15K-237, we find that there is a great deal of diversity in the similarity rankings between approaches. No pair of methods have an average Jaccard similarity exceeding 0.3125, which is the similarity between ComplEx and DistMult. The remaining approaches show little to no overlap, suggesting that each approach is modeling the similarities between predicates in entirely different ways. It is interesting to find that there are very few commonalities between the neural approaches, many of which utilize similar architectures. We also acknowledge that the KL-div approach has the least overlap with other approaches. Upon further investigation, we believe this is due to improper model specification. To highlight this issue, we selected the five most frequent predicates in FB15K-237. For each of these predicates, we computed the top five most similar predicates using each of the selected approaches. Overall, we found that the

KL-div method favored one particular predicate, namely

$$/location/hud\_foreclosure\_area/estimated\_number\_of\_mortgages,$$

as the most similar to all other predicates. We suspect that this predicate serves as a hub, where its frequency in the amount of triples is 'about average'. Clearly, this type of inference is incorrect and the representation of this predicate derived from the **KL-div** method dominates the predicate space. For other approaches, the similarities are more semantically interpretable– predicates related to acting and film are all found to be similar to one another, and similar patterns exist for sports related predicates and place/location predicates.

Table 5
FB15K-237: Average Jaccard similarities of ranked candidate predicates between approaches.

	KL-div	PF/IPF	ComplEx	ConvE	DistMult	RESCAL	RotatE	TransE
KL-div	n/a	0.0004	0.0048	0.0066	0.0084	0.0590	0.0060	0.0032
PF/IPF	·	n/a	0.2570	0.1619	0.2189	0.0723	0.20171	0.1946
ComplEx	·	·	n/a	0.1812	0.3125	0.0606	0.2258	0.2219
ConvE	·	·	·	n/a	0.1625	0.0564	0.1393	0.3639
DistMult	·	·	·	·	n/a	0.0553	0.1723	0.1754
RESCAL	·	·	·	·	·	n/a	0.0383	0.0445
RotatE	·	·	·	·	·	·	n/a	0.1615
TransE	·	·	·	·	·	·	·	n/a

Table 6
WN18RR: Average Jaccard similarities of ranked candidate predicates between approaches.

	KL-div	PF/IPF	ComplEx	ConvE	DistMult	RESCAL	RotatE	TransE
KL-div	n/a	0.24819	0.1010	0.3463	0.1677	0.2193	0.2896	0.2839
PF/IPF	·	n/a	0.1233	0.3311	0.4772	0.4285	0.3311	0.4321
ComplEx	·	·	n/a	0.0883	0.1010	0.1334	0.0883	0.1461
ConvE	·	·	·	n/a	0.3564	0.3365	0.2896	0.3474
DistMult	·	·	·	·	n/a	0.5003	0.3672	0.4484
RESCAL	·	·	·	·	·	n/a	0.3510	0.3906
RotatE	·	·	·	·	·	·	n/a	0.3654
TransE	·	·	·	·	·	·	·	n/a

6.2. Analysis of Triple Classification

In work currently under review, we show that the PTSS post-processing scheme leads to dramatic improvements in triple classification when compared to the previous state-of-the-art benchmark, Triple2vec [12]. In particular, we show a 67% improvement in Micro-F1 scores on WN18RR (.4008 to .6723) and 1247% improvement in Micro-F1 scores for FB15K-237 (.059 to .7949). These advances come from the ability of PTSS to leverage information already learned through the KGE methodologies serving as seed vectors as well as deviating from the random walk methodologies used in Triple2vec, which we believe are less applicable to sparse knowledge

graphs such as Freebase. Given these improvements, we ran the following experiments to test the efficacy of the predicate similarity function and exploit any further gains from alternate similarity measures. To summarize the results of the ninety experiments done on each dataset, we have compiled the following high level findings.

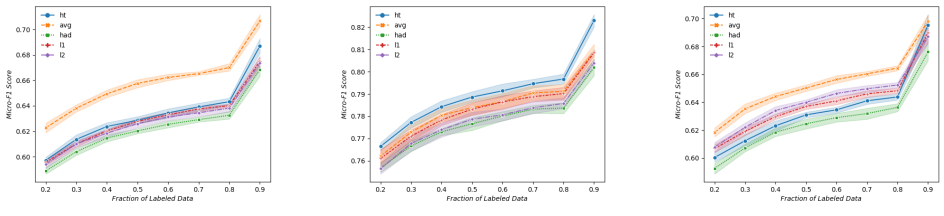
First, we identify the best performing models for each dataset, which we define as the maximum Micro-F1 score when training on 80% of the available triples. For FB15K-237, the best benchmark models is **RotatE_HT_EMB** with an average score of 0.725 and 0.754 for the low and high capacity models, respectively. This benchmark is improved upon when using the scores derived from **KL-div**, where the best model is **ConvE_HT_KLDIV** with average scores of 0.792 and 0.796. While we see an improvement when using the **KL-div** scores in conjunction with the PTSS scoring, the case is very different when comparing **PF/IPF**, where there is no clear individual model winner, with the best approach being **RotatE_HT_PFIPF** having scores of 0.485 and 0.513. Substituting **KL-div** for the baseline embedding similarity results in lifts of 9.24% and 5.57% for low and high capacity models, respectively.

For WN18RR, the findings are consistent, with the **KL-div** method giving best results, followed by **EMB** and **PF/IPF**. Using the **PF/IPF** scores gives best performance with the model **ConvE_HAD_PFIPF** with scores of 0.347 and 0.35. When using the **KL-div** features, the same base model and aggregation yields best performance, with **ConvE_HAD_KLDIV** having maximum scores of 0.522 and 0.545. Performance on **ConvE_HAD_EMB** resulted in scores of 0.397 and 0.5. This translates to a lift of 31.48% and 9% when moving from the embedding-based model to **KL-div** model for the low and high capacity models, respectively. The greater lift seen on the WN18RR dataset correlates back to the finding that the entropy of the predicate similarity matrix was greatest when using this approach. In the case of both datasets, including the **KL-div** results leads to models that improve on the baseline embedding models, suggesting that the information captured by this method is more reflective of predicate semantics. We present the full results for evaluation of triple classification when using **KL-div** in Figures 6.2 and 6.2.

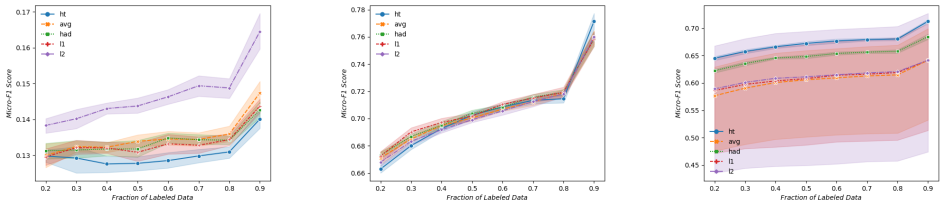
7. Related Work

We cover the requisite background on the knowledge graph embedding models tested in our experiments. For a full coverage of these techniques, please see [13]. These methods fall into three main groups: translation-based models, semantic-matching models, and graph-based models. The intuition behind translation-based models is to build vector representations of $\langle h, p, t \rangle$ such that $\mathbf{h} + \mathbf{p} \approx \mathbf{t}$. Model choices then depend on which space or spaces the entities and relations are embedded in as well as the scoring function used to help the model learn to differentiate between true triples from the graph and noise triples that do not reflect real-world facts. **TransE** [1] is the simplest of these models that embeds entities and predicates by using a distance function defined by $f_p(\mathbf{h}, \mathbf{t}) = -\|\mathbf{h} + \mathbf{p} - \mathbf{t}\|_{1/2}$. While this model is simple, it struggles to properly encode one to many triples, where a single relation may hold between a head entity and several tail entities. Semantic-matching models deviate away from the distance-based assumption and focus on using similarity-based scoring functions. These methods leverage dot-product-like scoring functions to measure angles between low-dimensional representations,

Figure 1: Micro F1 scores for using predicate fact distributions on FB15k-237 triple classification task

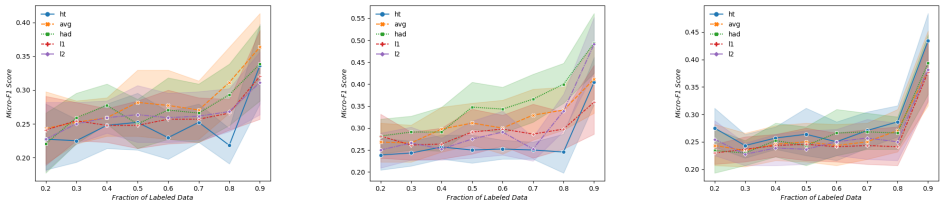


(a) ComplEx (b) ConvE (c) DistMult

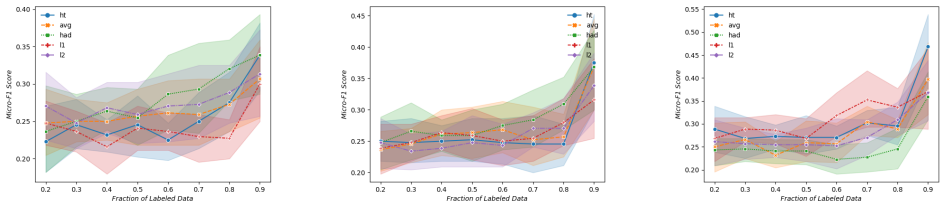


(d) RESCAL (e) RotatE (f) TransE

Figure 2: Micro F1 scores using predicate fact distributions on WN18RR triple classification task



(a) ComplEx (b) ConvE (c) DistMult



(d) RESCAL (e) RotatE (f) TransE

sometimes referred to as 'semantic energy' functions. The simplest of such models is **RESCAL** [10], which relies on a tensor representation of the underlying knowledge graph X, where each entry of the tensor $\mathbf{X}_{ijk} = 1$ if the fact is represented in the knowledge graph, otherwise zero. The tensor can then be decomposed into latent components, where each predicate-specific matrix P_k is a matrix of dimension $r \times r$ representing interactions between each corresponding component. In a simplification, **DistMult** [2] requires each P_k to be diagonal, reducing the parameters of the model while sacrificing some of its representational capacity. This reduction in capacity is especially felt when modeling anti-symmetric relations as interactions in these

diagonal matrices have no notion of directionality. To circumvent this issue, the **ComplEx** [7] model allows for the low-dimensional representations to live in the complex space $\mathbb{C}$. The **RotatE** approach also utilizes complex-valued representations, but it also models each individual relation as a rotation between the head and tail entities. This additional parameterization allows for increased learning capacity with the end goal of capturing anti-symmetry, inversion, and composition. The work of [8] takes this one step further, defining the **ConvE** model where entities interact through the convolution operator, introducing additional non-linearities to increase the capacity for learning complicated relational structures.

These models have been probed on the amount of *semantic* information they capture. The work of [3] finds that entity semantics are not universal; only a small subset of entities follow expected semantic patterns. While we agree that semantics are not the target of KGE models, our work shows that they may still be leveraged via weak supervision to reintroduce semantics to a triple representation. Our ablation of predicate similarity measures shows there is room for improvement on pure associative vector representations; our future research will continue to explore blended methodologies.

8. Conclusion

In this work, we introduce a novel method for quantifying the similarity between knowledge graph triples and utilize these scores to build neural representations of triples as first-class objects. We find that the information contained in traditional knowledge graph embeddings is not enough to build semantically grounded triple representations. Instead, we find that introducing additional measures of predicate relatedness, namely through the KL-divergence of fact distributions introduced by [5], leads to improvements when coupled with our methodology. Even with this gain, there is clear room for improvement, such as removing the tendency to identify predicate hubs detailed in Section 6.1. Future work will continue to explore the relationship between neural and distributional approaches and how they best combine for semantic knowledge graph triple representation.

References

[1] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: C. J. C. Burges, L. Bottou, Z. Ghahramani, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems (NeurIPS) 2013., 2013, pp. 2787–2795.

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

[3] N. Jain, J.-C. Kalo, W.-T. Balke, R. Krestel, Do embeddings actually capture knowledge graph semantics?, in: Eighteenth Extended Semantic Web Conference (ESWC) - Research Track, 2021.

[4] G. Pirrò, Building relatedness explanations from knowledge graphs, Semantic Web 10 (2019) 963–990.
[5] W. Chen, H. Zhu, X. Han, Z. Liu, M. Sun, Quantifying similarity between relations with fact distribution, CoRR abs/1907.08937 (2019). URL: http://arxiv.org/abs/1907.08937. arXiv:1907.08937.
[6] S. Broscheit, D. Ruffinelli, A. Kochsiek, P. Betz, R. Gemulla, LibKGE - A knowledge graph embedding library for reproducible research, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations, 2020, pp. 165–174.
[7] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, G. Bouchard, Knowledge graph completion via complex tensor factorization, Journal of Machine Learning Research (JMLR) 18 (2017) 1–38.
[8] T. Dettmers, M. Pasquale, S. Pontus, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of the 32th AAAI Conference on Artificial Intelligence, 2018, pp. 1811–1818. URL: https://arxiv.org/abs/1707.01476.
[9] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, 2019. arXiv:1902.10197.
[10] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, Omnipress, Madison, WI, USA, 2011, p. 809–816.
[11] A. Conneau, D. Kiela, SentEval: An evaluation toolkit for universal sentence representations, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), European Language Resources Association (ELRA), Miyazaki, Japan, 2018.
[12] V. Fionda, G. Pirrò, Learning triple embeddings from knowledge graphs, volume 34, 2020, pp. 3874–3881. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5800. doi:10.1609/aaai.v34i04.5800.
[13] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering PP (2017) 1–1.