Popularity Agnostic Evaluation of Knowledge Graph Embeddings

Aisha Mohamed ★ **Shameem A. Parambath** ★ **Zoi Kaoudi** ♦[*] **Ashraf Aboulnaga** ★

★Qatar Computing Research Institute, HBKU ♦Technische Universität Berlin

Abstract

In this paper, we show that the distribution of entities and relations in common knowledge graphs is highly skewed, with some entities and relations being much more popular than the rest. We show that while knowledge graph embedding models give state-of-the-art performance in many relational learning tasks such as link prediction, current evaluation metrics like `hits@`k and `mrr` are biased towards popular entities and relations. We propose two new evaluation metrics, `strat-hits@`k and `strat-mrr`, which are unbiased estimators of the true `hits@`k and `mrr` when the items follow a power-law distribution. Our new metrics are generalizations of `hits@`k and `mrr` that take into account the popularity of the entities and relations in the data, with a tuning parameter determining how much emphasis the metric places on popular vs. unpopular items. Using our metrics, we run experiments on benchmark datasets to show that the performance of embedding models degrades as the popularity of the entities and relations decreases, and that current reported results overestimate the performance of these models by magnifying their accuracy on popular items.

1 INTRODUCTION

Recently, a number of large-scale knowledge graphs like DBpedia [1], YAGO [29], WordNet [17], Freebase [3], and Google Knowledge Graph [25] have been created and deployed in many real world applications including search systems, virtual assistants, and question answering. A knowledge graph is a dataset of general relational facts about entities in the world in the form of $(subject, relation, object)$ triples. Subjects and objects represent the set of real-world entities (nodes) in the graph and each triple represents an edge of type $relation$ between the $subject$ and the $object$. While current knowledge graphs contain a huge amount of information with relatively high accuracy, they are still far from complete. Relational learning models reason over the rich semantic structure of knowledge graphs to add missing correct facts to the graph and detect incorrect facts to remove from it.

Embedding models for knowledge graphs have shown state-of-the-art performance in relational learning tasks such as link prediction (a.k.a. knowledge graph completion) and entity resolution [18, 24]. These models embed the entities and relations in a latent subspace and then use the embeddings to infer unobserved links between the entities. To predict whether a missing triple (e_i, r_k, e_j) is true, a score function $f((e_i, r_k, e_j); \dots)$ is calculated based on the embeddings of the constituent entities and relation. The score indicates the confidence of the model that the entities e_i and e_j are related by r_k [23]. The main evaluation criteria for these models is: given a triple with a missing subject or object entity, how well does the model rank the correct missing entity compared to other entities in the graph?

A fundamental problem with the current entity ranking evaluation metrics is their bias towards popular entities and relations. There is a popularity bias in knowledge graphs (many facts about few popular entities and few facts about most of the other long-tail entities) and the evaluation metrics do not correct for this bias in the data. The popularity bias in knowledge graphs can be attributed to the fact that most of these graphs are automatically constructed in a best-effort way from online sources that suffer from intrinsic biases (e.g., Wikipedia). This bias towards popular entities and relations in the web sources used in knowledge graph construction is reflected and amplified in the created knowledge graphs.

[*]Work done while at QCRI.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

Current evaluation metrics like $\texttt{hits@}k$ and $\texttt{mrr}$ calculate the accuracy of a model as follows:

$$^{1}/_{|Test|} \sum_{(e_i, r_k, e_j) \in Test} \frac{\texttt{metric}(e_i, r_k, ?) + \texttt{metric}(?, r_k, e_j)}{2} \quad (1)$$

That is, for every test triple (e_i, r_k, e_j), how well does the model predict the subject if it is missing and the object if it is missing? This per-triple metric is averaged over the entire test set. In Equation 1, the metric implicitly scales the contribution of each entity or relation to the overall accuracy of the model proportionally to its frequency in the test data (popularity), since the summation ranges over all test triples. Hence, such metrics provide a biased estimation of the accuracy of the model since they under-emphasize the value of predictions about the unpopular, long-tail entities and relations. Using these biased metrics can lead to the development of models that perform well on popular entities and relations and ignore the rest. This contradicts the principal motivation for using embedding models, which is to propagate information through the structure of the graph to unfamiliar entities and relations and use this information to infer new knowledge that cannot be easily extracted by on-line text mining or inferred using simple graphical models.

Though the problem of long-tail entities and relations is well-studied in many machine learning contexts (e.g., minority classes in classification and popularity bias in recommendation systems), the traditional solutions used to de-bias the evaluation metrics, such as micro-averaging, do not directly apply to knowledge graph embeddings. The training instance in knowledge graph embeddings is the *triple*, which has three components: a subject entity, an object entity, and a relation. In most real knowledge graphs, popular entities do not co-occur with popular relations in the same triple (e.g., it is possible for only the subject to be popular while the object and relation are not popular). Hence, a triple cannot be characterized in its entirety as popular or unpopular. Thus, a simple re-weighting of triples cannot address popularity bias in this context.

Another problem with traditional approaches to debiasing evaluation metrics is that these approaches quantify the accuracy of a model by one number that shows accuracy under a specific re-weighting (e.g., equal weights to all items). An alternate, and better, approach is to use a series of numbers (together making a trend line) showing the accuracy of the model as the weight is gradually shifted from the most popular items towards the least popular items. The unpopular items in a knowledge graph can be divided into two sub-categories: (1) distant-tail items that occur in so few triples (sometimes one triple) that reliable predictions about them are unexpected, and (2) long-tail items that occur in enough triples that a good model can learn a meaningful embedding of these items. Measuring accuracy via a trend line would be helpful in discriminating between models that do well on most items and perform poorly only on the distant-tail items, and models that perform well only on the most popular items and perform poorly on the long-tail and distant-tail items, the majority of the items in the knowledge graph. A good embedding model should perform well on popular and long-tail items.

Guided by these insights, we propose a new class of evaluation metrics, $\texttt{strat-hits@}k$ and $\texttt{strat-mrr}$ (*stratified Hits@k* and *stratified MRR*), and use it to show that state-of-the-art embedding models are biased towards popular entities and relations. These metrics can expose the popularity bias in the embedding models. Our metrics do not assume that popular entities co-occur with popular relations in the same triples, and so can be used with knowledge graphs with any level of correlation between entity and relation popularity. We use our metrics to show that the accuracy of embedding models decreases on triples representing facts about unpopular entities or relations. This can be attributed to the training procedure of these models, since it iterates over all the triples once every epoch regardless of the popularity bias in the data. Popular items get more focus during training. Hence, the popularity bias in the input knowledge graphs makes the inference of new facts that include unpopular entities and relations more challenging, even though these facts are more valuable for enriching the graph compared to facts about popular items. Note that, while prior work has criticized the benchmark datasets used for training and evaluation of knowledge graph embedding [24, 30], no work has investigated the impact of popularity bias on the evaluation metrics.

The contribution of this paper lies in highlighting a common but previously unexplored problem that affects knowledge graph embedding models and presenting a novel metric to quantify the effect of this problem on accuracy. We introduce the knowledge graph embedding models that we use in Section 2. In Section 3, we empirically show that in real knowledge graphs: (1) there is a selectivity bias towards popular entities and relations, and (2) popular entities and relations are not correlated (i.e., do not co-occur in the same triple). We then show that current evaluation metrics are mathematically biased towards popular entities and relations and propose our new evaluation metrics to correct this bias in Section 4. We present an experimental evaluation in Section 5, showing that current evaluation of embedding models over-estimates their accuracy; these models are accurate only on popular entities and relations. We discuss related work in Section 6 and conclude in Section 7.

2 PRELIMINARIES: KNOWLEDGE GRAPH EMBEDDING MODELS

Let $\mathcal{E} = \{e_1, e_2, \dots, e_N\}$ be the set of entities and $\mathcal{R} = \{r_1, r_2, \dots, r_K\}$ be the set of all relation types in a knowledge graph. A triple is of the form (e_i, r_k, e_j), where $e_i \in \mathcal{E}$ is the subject, $e_j \in \mathcal{E}$ is the object, and $r_k \in \mathcal{R}$ is the relation between them. Let $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ be the set of all possible triples. A knowledge graph $\mathcal{G} \subseteq \mathcal{T}$ is a subset of all possible triples with $N = |\mathcal{E}| \geq 2$ entities, $K = |\mathcal{R}| \geq 1$ relations, and $M = |\mathcal{G}|$ triples. $\mathcal{G}$ can be represented by a third-order binary tensor $\underline{\mathbf{Y}} \in \{0,1\}^{N \times N \times K}$ where the random variable $y_{ijk} = 1$ iff $(e_i, r_k, e_j) \in \mathcal{G}$, where i, j, and k are the positions of e_i, e_j, and r_k in the tensor, respectively.

Given a knowledge graph $\mathcal{G}$, *link prediction* is the problem of finding the probability that any triple $t \in \mathcal{T}$ exists in the graph. By setting a threshold on the probability, one can determine whether a triple is true or not (and hence assign it a label from $\{-1, 1\}$). Link prediction is equivalent to estimating the joint probability distribution $P(\underline{\mathbf{Y}})$ of correctness of all triples in $\mathcal{T}$ given the set of labeled observed triples $D \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \{-1, 1\}$. We denote as $D^+ \subseteq D$ the set of positive labeled triples (true triples) and $D^- \subseteq D$ the set of negative labeled triples (false triples). It holds that $D = D^+ \cup D^-$. The positive triples in D^+ are a subset of the triples in $\mathcal{G}$. Negative triples are constructed by different heuristics as we discuss later.

Knowledge graph embedding models learn an embedding of all entities and relations in the graph in a low-dimensional space. These models predict the existence of a triple $t = (e_i, r_k, e_j)$ via a scoring function $f(t; \Theta)$ which represents the model's confidence that t exists given the model parameters Θ. Θ consists of the learned latent representations of the entities e_i, e_j and relation r_k of t. We denote these representations as $\mathbf{e}_i \in \mathbb{R}^l$, $\mathbf{e}_j \in \mathbb{R}^l$, and $\mathbf{r}_k \in \mathbb{R}^l$, respectively, where $l \in \mathbb{N}$ is the size of the model. Embedding models aim to represent the semantics of each entity and relation in its latent representation and to use this embedding to correctly predict the scores of true and false triples. Below, we outline the scoring functions of the four popular embedding models that we consider in this paper.

TransE: TransE [4] is a translation-based model inspired by the Word2vec algorithm [16]. It represents a relation as a translation operation on the entities and uses a scoring function that measures the distance of the two entities with respect to the relation of the triple: $f_t^{TransE} = -d(\mathbf{e}_i + \mathbf{r}_k, \mathbf{e}_j)$ where $d(x, y)$ can be any distance measure, e.g., L1 or L2 norm.

DistMult: DistMult [2] can be seen as a more compact and less expressive variant of RESCAL, the first factorization-based embedding model [20]. It adds a diagonality constraint on the relation matrix and can thus, model only symmetric relations. Its scoring function is: $f_t^{DistMult} = \mathbf{e}_i^T diag(\mathbf{r}_k)\mathbf{e}_j$.

HolE: Leveraging the idea of associative memory, HolE [19] uses a circular correlation operation between the two entities' vectors in its scoring function: $f_t^{HolE} = \mathbf{r}_k^T(\mathbf{e}_i \star \mathbf{e}_j)$ where $(\mathbf{e}_i \star \mathbf{e}_j)_k = \sum_{t=1}^{l} e_{it} e_{j((k+t-2 \bmod l)+1)}$.

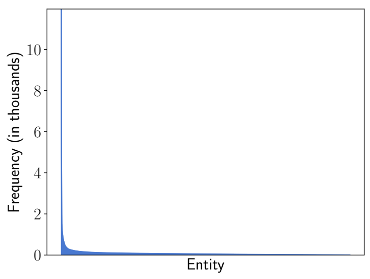
ComplEx: ComplEx [31] extends DistMult by using complex numbers and the Hermitian dot product. Its scoring function is: $f_t^{ComplE} = Real(\mathbf{e}_i^T diag(\mathbf{r}_k)\mathbf{e}_j)$ where $Real()$ is a function that returns the real part of a complex number. ComplEx has been shown to be equivalent to HolE.

3 POPULARITY BIAS IN KNOWLEDGE GRAPHS

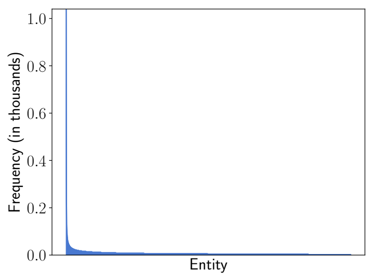
Most of the popular knowledge graphs (e.g. DBpedia, Wikidata, and YAGO) are automatically constructed from web sources like Wikipedia or by mining text documents using information extractors contributed by developers [1, 29]. These web sources are subject to popularity bias. This is attributed to the fact that popular knowledge is readily available in many online sources and more often completed by users of these platforms, as compared to unpopular or rare knowledge. Hence, facts about popular entities are easier to extract with high accuracy compared to the same types of facts about unpopular entities. Moreover, simple facts that describe simple relations between entities, such as the *type* relation between each entity and its class, are available for more entities compared to the more complex relations that describe complicated and meaningful interactions between entities in the graph. For example, though Nollywood is the second biggest movie industry in the world in terms of the number of movies produced, second only to Bollywood, Wikipedia contains information (both partial and full) regarding only 83 Nollywood movies, whereas full information regarding thousands of Hollywood movies can be found in Wikipedia. Such data bias is inherited and can be detected in almost all the commonly used knowledge graphs. As argued by [6], the problem is amplified by the use of crowdsourcing in knowledge graph construction.

Table 1: Statistics of Benchmark Datasets

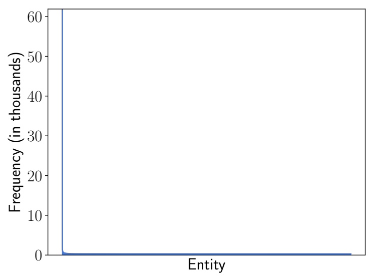
	NUMBER OF TRIPLES (train/validation/test)	ALL POPULAR	ALL UNPOPULAR	MIX OF POPULAR AND UNPOPULAR
FB15K	483K / 50K / 59K	70K	45K	477K
WN18	141.4K / 5K / 5K	7.6K	26.6K	117K
YAGO3-10	1.079M / 5K / 5K	7K	120K	962K



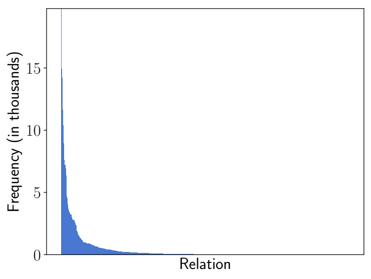
(a) FB15K Entities



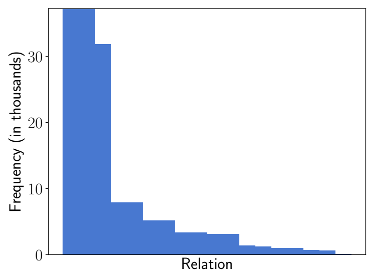
(b) WN18 Entities



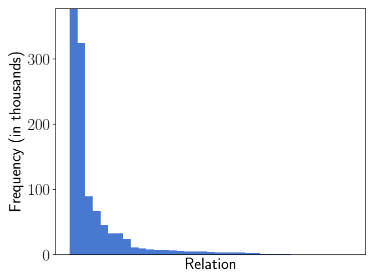
(c) YAGO3-10 Entities



(d) FB15K Relations



(e) WN18 Relations



(f) YAGO3-10 Relations

Figure 1: Frequency Distribution of Entities and Relations in the Triples of Benchmark Datasets

3.1 POPULARITY BIAS IN BENCHMARK DATASETS

Researchers evaluate embedding models on small benchmark datasets that are constructed from the most frequently occurring entities and relations in their original knowledge graphs rather than the full knowledge graphs. This is because representing huge real-world knowledge graphs that contain hundreds of millions of entities and tens of thousands of relations with an embedding dimensionality that can capture all the information in such graphs is still computationally infeasible on commonly available machines [21]. Three of the most popular benchmark datasets are FB15K [4], extracted from FreeBase, WN18 [4], extracted from WordNet, and YAGO3-10 [7], extracted from YAGO. Each of these datasets is divided into subsets for training, validation, and testing. The statistics of these datasets are given in Table 1.

Figure 1 presents histogram plots for the frequency of entities and relations for these three benchmark datasets. While these datasets were constructed from the most frequently occurring entities and relations, they clearly still exhibit a skewed, power-law distribution even within these most frequent entities and relations. The frequency distribution of the entities is more skewed compared to relations as the number of entities is much larger than the number of relations in all the knowledge graphs.

3.2 ENTITY POPULARITY VS. RELATION POPULARITY

Popularity bias, a.k.a. data imbalance, is a very common problem that has traditionally been tackled in machine learning by one of the following methods: artificially resampling the dataset, adjusting the cost associated with errors to be biased towards minority classes, or applying feature selection to reduce the data imbalance [14]. Applying these methods to knowledge graphs is challenging due to the structural differences between tabular and graph data. The training instance for knowledge graph embedding models is the triple. Simply applying these methods assumes that each triple can be characterized as either popular or unpopular. However, this is not the case since the popularity of the subject and object entities and the relation in the same triple are not necessarily correlated.

To illustrate this on the benchmark datasets, Table 1 splits the triples in the datasets based on the popularity of their subject, object, and relation. We assume a $(90/10)\%$ split where the most popular 10% of entities and relations are considered popular and the rest are considered unpopular. The third column of the table shows the number of triples in which the subject, predicate, and relation are all popular. The fourth column shows the number of triples in which the subject, predicate, and relation are all unpopular. The last column of the table shows the number of triples containing a mix of popular and unpopular subject, object, and/or relation. The bulk of the triples in all the datasets is in this last column. This demonstrates that identifying popular or unpopular subjects, objects, and relations is more meaningful than identifying popular or unpopular triples. We tried different thresholds for popularity (e.g., 20% and 25%) and the same phenomenon persists. Another measure of this phenomenon is the correlation between the popularity of the subjects and relations in the triples, and between the popularity of the objects and relations. These correlations are all close to zero. The highest in absolute value is between objects and relations in YAGO3-10, at -0.3.

4 POPULARITY STRATIFIED EVALUATION METRICS

The popularity bias in knowledge graphs can affect the training process of knowledge graph embedding models and can lead to models that perform well only on popular entities or relations. Therefore, the evaluation process should take the popularity bias into account and use an evaluation metric that considers both popular and unpopular items. In this section, we define the stratified $\texttt{hits@}k$ and stratified $\texttt{mrr}$ metrics, which generalize the popular $\texttt{hits@}k$ and $\texttt{mrr}$ metrics by introducing popularity weights on entities and relations.

A common approach used in the literature to reduce the popularity bias in recommendation systems is to reweight the recommendations based on the popularity of the item [27]. In knowledge graphs, as popularity bias can happen in entities and in relations, we need to reweight both entities and relations. We assume that the probability that a triple (e_i, r_k, e_j) appears in a knowledge graph depends on the popularity of the constituent entities and relation. Furthermore, we assume that the probabilities of entities and relations occurring in the graph are independent of each other. We define the popularity of an entity or relation x by the number of times it appears in the graph either as a subject, an object, or a relation. Let $N(x)$ be the popularity of x in the complete knowledge graph. In practical settings, $N(x)$ cannot be computed due to many missing edges in the observed knowledge graph. The empirical probability of observing a triple containing x is $\hat{p}(x) = \hat{N}(x)/N(x)$ where $\hat{N}(x)$ is the number of times x appears in the observed knowledge graph.

From the empirical evidence provided in Figure 1, we can see that the entities and relations follow a power-law distribution. In case of power-law distributed variables, we can say that $\hat{p}(x) \propto [N(x)]^\alpha$ where $\alpha \in \Re_+$ is the tail index. A smaller value of α indicates that the distribution contains more extreme values with a very heavy tail (many unpopular items) and a larger value of α indicates a weak tail with few unpopular items. The probability of entity or relation x being observed is:

$$\begin{aligned} \hat{p}(x) &= c[N(x)]^\alpha \text{ (scale invariance of power law)} \\ &= c[\frac{\hat{N}(x)}{\hat{p}(x)}]^\alpha \text{ (using the definition of } \hat{p}) \\ [\hat{p}(x)]^{\alpha+1} &= c[\hat{N}(x)]^\alpha \text{ (re-arranging the terms)} \\ \hat{p}(x) &\propto [\hat{N}(x)]^\beta \text{ (scale invariance; } \beta = \alpha/\alpha+1) \end{aligned} \quad (2)$$

The two most common evaluation metrics used in the knowledge graph literature are: $\texttt{hits@}k$ and $\texttt{mrr}$. For a triple (e_i, r_k, e_j), $\texttt{hits@}k(e_i, r_k, e_j) =$

$$\frac{\mathbb{1}[e_j \in \texttt{top-k}(e_i, r, *)] + \mathbb{1}[e_i \in \texttt{top-k}(*, r, e_j)]}{2} \quad (3)$$

where $\mathbb{1}$ is the indicator function, $\texttt{top-k}(e_i, r_k, *)$ is the set of top k ranked entities for the relation r_k with e_i as the subject entity, and $\texttt{top-k}(*, r_k, e_j)$ is the similar set with e_j as the object entity. The mean reciprocal rank for the same triple $\texttt{mrr}(e_i, r_k, e_j) =$

$$\frac{1}{2}\left(\frac{1}{rank(e_j)\texttt{in}(e_i, r_k, *)} + \frac{1}{rank(e_i)\texttt{in}(*, r_k, e_j)}\right) \quad (4)$$

where $rank(e_j)\texttt{in}(e_i, r_k, *)$ is the rank of e_j in the set of ranked predictions for the relation r_k with e_i as the subject entity, and conversely for $rank(e_i)\texttt{in}(*, r_k, e_j)$.

For a given relation r, we define $\texttt{hits}_\texttt{r}\texttt{@}k$ and $\texttt{mrr}_\texttt{r}$ for relation r, which gives equal weight to all pairs of entities as:

$$\begin{aligned} \texttt{hits}_{\texttt{r}}@k &= \frac{1}{|\mathcal{E}(r)|} \sum_{(e_i,e_j)\in\mathcal{E}(r)} \texttt{hits}@k(e_i, r, e_j) \\ \texttt{mrr}_{\texttt{r}} &= \frac{1}{|\mathcal{E}(r)|} \sum_{(e_i,e_j)\in\mathcal{E}(r)} \texttt{mrr}(e_i, r, e_j) \end{aligned} \tag{5}$$

where $\mathcal{E}(r)$ is the set of corresponding entity pairs that occur with r in the test data. The overall $\texttt{hits}@k$ or $\texttt{mrr}$ score for the dataset is defined as:

$$\texttt{metric} = \frac{\sum_{r\in\mathcal{R}} \texttt{metric}_{\texttt{r}} \times |\mathcal{E}(r)|}{\sum_{r\in\mathcal{R}} |\mathcal{E}(r)|} \tag{6}$$

where $\texttt{metric}$ is $\texttt{hits}@k$ or $\texttt{mrr}$.

As pointed out earlier, a common and well established approach for de-biasing the popularity in the estimated quantity is to re-weight the predictions [11, 27]. Modifying Equation 5 to take into account the weight of the entities, we define the stratified metrics as $\texttt{strat-hits@k}_r$ and $\texttt{strat-mrr}_r$ for relation r as follows:

$$\begin{matrix}\texttt{strat-}\\ \texttt{metric}_r\end{matrix} = \frac{1}{|\mathcal{E}(r)|} \sum_{(e_i,e_j)\in\mathcal{E}(r)} \begin{matrix}\texttt{strat-}\\ \texttt{metric}\end{matrix}(e_i, r, e_j) \tag{7}$$

where $\texttt{strat-hits}@k(e_i, r_k, e_j) =$

$$\frac{w_{e_j}\mathbb{1}[e_j \in \texttt{top-k}(e_i, r, *)] + w_{e_i}\mathbb{1}[e_i \in \texttt{top-k}(*, r, e_j)]}{w_{e_i} + w_{e_j}} \tag{8}$$

and $\texttt{strat-mrr}(e_i, r_k, e_j) =$

$$\frac{1}{w_{e_i} + w_{e_j}} \left(\frac{w_{e_j}}{rank(e_j)\texttt{in}(e_i,r_k,*)} + \frac{w_{e_i}}{rank(e_i)\texttt{in}(*,r_k,e_j)} \right) \tag{9}$$

where w_{e_i} and w_{e_j} are the weights of entities e_i and e_j, respectively.

Finally, we define $\texttt{strat-hits}$ and $\texttt{strat-mrr}$ for the entire dataset as:

$$\texttt{strat-metric} = a \sum_{r} w^r \times \texttt{strat-metric}_r \tag{10}$$

where w^r is the weight of relation r and a is a normalization factor. Following [26], we assume $w^r \propto s(r)$ and $w_{e_i} \propto s(e_i)$ where $s(x) = \frac{1}{\hat{p}(x)} \propto \frac{1}{[\hat{N}(x)]^\beta}$ and the weights are normalized such that the final $\texttt{strat-metric}$ score $\in [0, 1]$. Note that there are separate β parameters for entities and relations, respectively, β_e and β_r. With the normalization constant, our final equation takes the form:

$$\texttt{strat-metric} = \frac{\sum_{r\in\mathcal{R}} w^r \times \texttt{strat-metric}_r}{\sum_{r\in\mathcal{R}} w^r} \tag{11}$$

We note that $\texttt{hits}@k$ applied to a single triple is exactly the same as the recall metric used in information retrieval. Following the expected utility maximization based definition of recall [22], one can see that $\texttt{strat-recall}$ [26] and $\texttt{strat-hits}$ are proportional to the true positive rate and thus equivalent. Since $\texttt{strat-recall}$ on the observed data provides a nearly unbiased estimate of the true recall [26], we can argue that $\texttt{strat-hits@k}$ is an unbiased estimate of $\texttt{hits}@k$.

Macro-averaging vs. Micro-averaging: In the classification literature, the *macro-average* evaluation metric gives equal weight to each class when computing the classification accuracy, whereas the *micro-average* gives equal weight to each instance or classification decision. Thus, micro-average is a measure of the classifier's overall performance on the most frequently occurring classes while macro-average gives a sense of the model's performance by giving equal weight to each class. The commonly used $\texttt{hits}@k$ and $\texttt{mrr}$ metrics can be viewed as micro-average metrics of the performance of the model with respect to relations and are dominated by the most popular relations. A macro-average would find the average of $\texttt{hits}_{\texttt{r}}@k$ or $\texttt{mrr}_{\texttt{r}}$ for all the relations giving equal weight to the relations regardless of their popularity. A large difference between the two metrics indicates a large popularity bias.

Effect of the Hyperparameters β_e and β_r: The hyperparameter β we have defined in $\texttt{strat-hits}@k$ controls the weighting factor for entities and relations. When $\beta = -1$, the weight of each item is proportional to the frequency of that item. Thus, the metric is biased towards popular items. As we increase β, we shift the focus more towards unpopular items. For $\beta \in [-1, 0)$, the weight of each item is proportional to the frequency of that item where higher β values downgrade all the weights and hence decrease the variance between popular and unpopular items. A value $\beta = 0$ gives equal weight to the items. For $\beta \in (0, 1]$, the weight of each item is inversely proportional to its frequency and higher beta values increase the variance between popular and unpopular weights. Plotting the curve of $\texttt{strat-hits}@k$ vs. β using an equally separated range of values of β gives valuable insights into the performance of a model. For example, models that perform well on popular and moderately unpopular items will be differentiated from models that perform well only on very popular items.

A very useful feature of these evaluation metrics is that they can be used to focus on popularity bias in either entities or relations by fixing one of the β values at 0 and varying the other one. To evaluate of the susceptibility of a model to popularity bias in, say, entities, we fix β_r at 0 and vary β_e from -1 to 1.

Notice that some specific values of β result in commonly used biased performance metrics, as stated in the following two lemmas.

Lemma 1 *For $\beta_r = -1$ and $\beta_e = 0$,* `strat-hits@`k *calculates the commonly used micro-averaged* `hits@`k *and* `strat-mrr` *calculates the commonly used micro-averaged* `mrr` *.*

Lemma 2 *For $\beta_e = \beta_r = 0$,* `strat-hits@`k *calculates the macro-averaged* `hits@`k *and* `strat-mrr` *calculates the macro-averaged* `mrr`*.*

5 EXPERIMENTAL EVALUATION

In this section we use our `strat-hits@`k and `strat-mrr` metrics to experimentally evaluate the impact of popularity bias on the performance of state-of-the-art knowledge graph embedding models.

Datasets: We use two popular link prediction benchmarks: FB15K and YAGO3-10 (Table 1). FB15K is a subset of Freebase [3], a knowledge graph that contains general information about the world, and YAGO3-10 is a subset of YAGO3 [15], a knowledge graph constructed from Wikipedia. YAGO3-10 was constructed from entities that have at least 10 relations each. We choose these two datasets as they are big enough and contain enough relations and entities to adequately evaluate the effect of popularity bias on model accuracy.

Experimental Setup: We evaluate the four embedding models described in Section 2: TransE, DistMult, HolE, and ComplEx. These embedding models contain translational and factorization based models and have shown competitive performance on most of the relational learning tasks. We use the implementation provided by AmpliGraph [5], an open source Python library that uses TensorFlow to implement knowledge graph embeddings, the loss functions, and optimizers commonly used to optimize them. We minimize the regularized logistic loss as described in [24], which has been shown to improve results significantly compared to the pair-wise ranking loss [13, 31]. We use the Adam optimizer as it requires substantially fewer iterations to converge compared to Adagrad [10]. We performed a grid search over the hyperparameters to maximize the filtered mean reciprocal rank (MRR) on the validation data. The grid search was done over the following range of hyperparameters: embedding size $d \in \{150, 200, 300, 350, 400\}$, batch size $\in \{50, 64, 100\}$, learning rate $\in \{0.001, 0.0005, 0.0001, 0.00005\}$, and number of negative examples $\eta \in \{10, 20, 30, 50\}$. We trained all models up to 2000 epochs.

Generation of Negative Examples: We use the standard approach for generating negative examples proposed in [4]: Let $\mathcal{D}^+$ be the set of positive training examples in the dataset. For each triple (s, p, o) in $\mathcal{D}^+$, we generate η negative triples by replacing the subject s or the object o with a random entity and filter the correct triples from these generated examples. Thus, the set of negative examples $\mathcal{D}^-$ contains the triples:

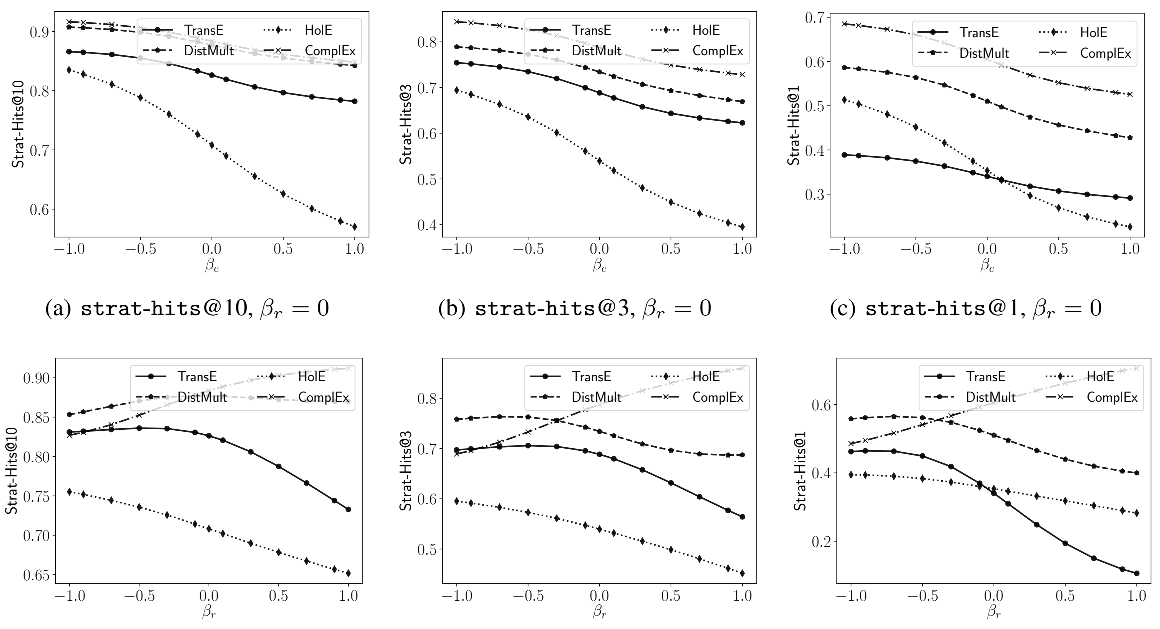
$$\{(s', p, o) | \; s' \in \mathcal{E} \wedge \; s' \neq s \; \wedge (s, p, o) \in \mathcal{D}^+\} \cup \\ \{(s, p, o') | \; o' \in \mathcal{E} \wedge \; o' \neq o \; \wedge (s, p, o) \in \mathcal{D}^+\}$$

The training set is $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. Following the link prediction literature, we randomly sample a different set of negative examples of size $\eta \times N$ from $\mathcal{D}^-$ in each epoch during training.

Evaluation Protocol: Following previous works on knowledge graph embedding models, we use the link prediction task for evaluation. For each true triple (e_i, r_k, e_j) in the test dataset, we replace e_i and e_j with each entity $e \in \mathcal{E}$ and score both the true and the corrupted triples using the model. Then we sort the triples by the value of their scores with higher scores ranked first. We compute the stratified `mrr` and stratified `hits@`k for $k \in \{1, 3, 10\}$ according to the equations described in Section 4. To show the effect of popularity bias on the accuracy of the model, we report the stratified metrics for gradually increasing values of $\beta \in [-1.0, 1.0]$ for both the entities and the relations.

Results: Figures 2 and 3 show the `strat-hits@`k of the four knowledge graph embedding models on FB15K and YAGO3-10, respectively, while varying the β hyperparameters that control the bias towards popular entities and relations. In sub-figures (a)–(c), we fix the weight of all relations to 1 by setting $\beta_r = 0$. We then plot `strat-hits@`k by varying β_e. The leftmost point in each figure, with $\beta_e = -1$, focuses on popular entities. As we move towards the right and increase β_e, we shift the focus of the metric towards less popular entities. We observe from the plots that the performance of all embedding models degrades as β_e increases. This means that knowledge graph embedding models derive their accuracy from doing well on popular entities while ignoring less popular ones. However, the less popular entities are often the more interesting ones in a link prediction task. Our `strat-hits@`k metric exposes this deficiency and provides a simple way to decide how much to focus on popular entities vs. less popular ones.

In sub-figures (d)–(f) we fix the weight of all entities to 1 by setting $\beta_e = 0$. We then plot `strat-hits@`k by varying β_r. Recall from Lemma 1 that the leftmost point on each figure, with $\beta_r = -1$, corresponds to the micro-averaged `hits@`k, which is the metric used in all knowledge graph embedding papers. The accuracy at this point is derived from performance on the popular relations. As we increase β_r, all knowledge graph embedding models



(a) `strat-hits@10`, $\beta_r = 0$ (b) `strat-hits@3`, $\beta_r = 0$ (c) `strat-hits@1`, $\beta_r = 0$

(d) `strat-hits@10`, $\beta_e = 0$ (e) `strat-hits@3`, $\beta_e = 0$ (f) `strat-hits@1`, $\beta_e = 0$

Figure 2: `strat-hits@`k of Knowledge Graph Embedding Models on FB15K

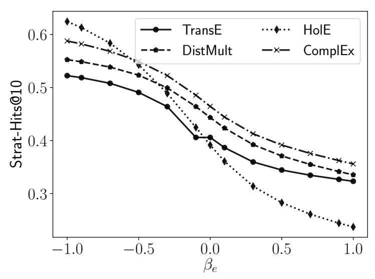
except ComplEx on FB15K suffer a degradation in performance. ComplEx performs better for unpopular relations. As before, we see that these state-of-the-art knowledge graph embedding models derive their accuracy in FB15K from doing well only on popular items. On YAGO3-10, the performance of the models degrades as the focus on popularity decreases, then it improves again. This means that the accuracy of the models is high on the very popular and the very unpopular relations and lower in the middle. This is rarely the desired performance of knowledge graph embedding models, since the most valuable predictions are the ones in the middle. However, we suspect that this behavior is specific to YAGO3-10 because it contains a small number of relations: YAGO3-10 contains only 37 relations, hardly representative of large real-world knowledge graphs. With more relations, the trend would likely be similar to FB15K.

Figure 4 shows the `strat-mrr` on the same two datasets. Sub-figures (a) and (c) fix $\beta_r = 0$ and vary the β_e on FB15K and YAGO3-10, respectively. Sub-figures (b) and (d) fix $\beta_e = 0$ and vary β_r on FB15K and YAGO3-10, respectively. The results show the same trends as the `strat-hits@`k figures, which confirms the popularity bias in knowledge graph embeddings and the ability of our metrics to consistently capture it.

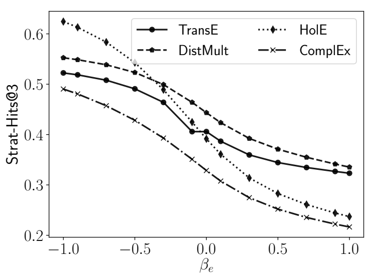
This clear bias in embedding models can be explained by the training procedure of these models. During training, the popular entities and relations have more background and context information available in the training data and get more focus during optimization. Also, The popular entities and relations occur in more triples and get updated more frequently. Hence, the model infers new facts about them with higher accuracy while unpopular entities and relations get overlooked.

6 RELATED WORK

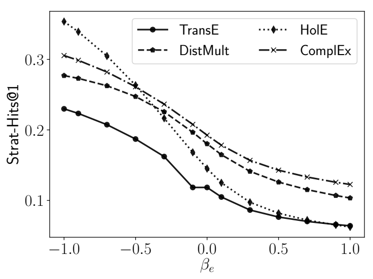
Recently, multiple works have discussed the inherent biases in knowledge graphs and their impact on relational learning tasks. Janowicz et al. [9] argue that cultural, geographical, and statistical biases in knowledge graphs can arise from the source of the data, its ontology, or the reasoning and rule learning systems used to enrich the graphs. Demartini [6] argues that collaborative crowdsourcing methods adopted in knowledge graph creation introduce the implicit biases of the contributors into the knowledge graph. Stepanova et al. [28] state that in rule mining systems, the popularity bias might lead to the extraction of erroneous biased rules. Li et al. [12] propose new evidence-collecting techniques to improve knowledge verification for long-tail entities that are not supported by enough information in the graph. Guo et al. [8] argue that triple-level learning in knowledge graph embeddings gives limited attention to long-tail entities, and thus their embeddings suffer from low expressiveness.



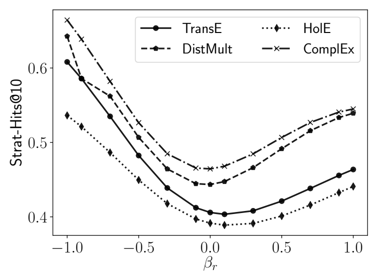
(a) `strat-hits@10`, $\beta_r = 0$



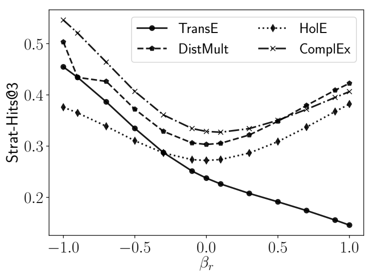
(b) `strat-hits@3`, $\beta_r = 0$



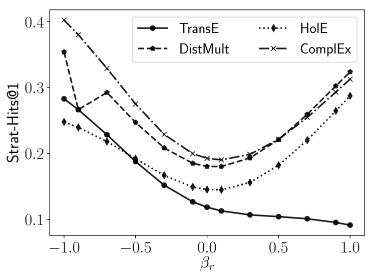
(c) `strat-hits@1`, $\beta_r = 0$



(d) `strat-hits@10`, $\beta_e = 0$

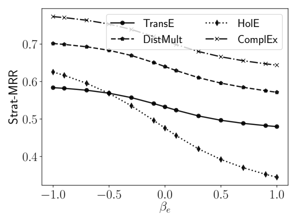


(e) `strat-hits@3`, $\beta_e = 0$

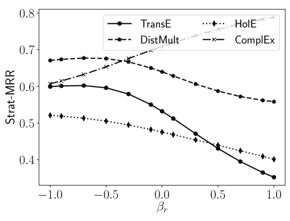


(f) `strat-hits@1`, $\beta_e = 0$

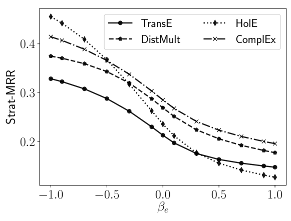
Figure 3: `strat-hits@`k of Knowledge Graph Embedding Models on YAGO3-10



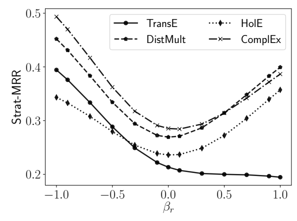
(a) FB15K, $\beta_r = 0$



(b) FB15K, $\beta_e = 0$



(c) YAGO3-10, $\beta_r = 0$



(d) YAGO3-10, $\beta_e = 0$

Figure 4: `strat-mrr` of Knowledge Graph Embedding Models on FB15K and YAGO3-10

They propose using path-level embeddings to improve performance on entity alignment and knowledge graph completion especially on long-tail entities but they do not quantify this improvement. Our work is the first to discuss bias in the evaluation metrics. Our proposed metrics offer unbiased estimations of accuracy and encourage development of unbiased embedding models.

7 CONCLUSION

In this paper, we study the effect of popularity bias on the performance of knowledge graph embedding models. We observe that knowledge graphs have a skewed popularity distribution for entities and relations, and the popularity of entities and relations is not necessarily correlated. We note that this popularity bias can have a detrimental effect on the training of knowledge graph embedding models and is not captured by current evaluation metrics. We propose the stratified Hits@k and stratified MRR metrics, which evaluate the accuracy of models on both popular and unpopular items, and have tuning parameters β that control the focus on popular vs. unpopular items. Using these metrics, we demonstrate that recent knowledge graph embedding models are indeed biased towards popular items, and we quantify this bias. Thus, we provide useful evaluation metrics that subsume the currently used ones and enable better understanding of the accuracy of embedding models on the long tail of the frequency distribution. Our future work is to use our novel evaluation metrics as a starting point for developing knowledge graph embedding models that do not suffer from popularity bias.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. 2007.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795, 2013.

[4] Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nick McCarthy, and Pedro Tabacof. AmpliGraph: A Library for Representation Learning on Knowledge Graphs, 2019.

[5] Gianluca Demartini. Implicit bias in crowdsourced knowledge graphs. In *WWW*, pages 624–630, 2019.

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

[7] Lingbing Guo, Zequn Sun, and Wei Hu. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*, pages 2505–2514, 2019.

[8] Krzysztof Janowicz, Bo Yan, Blake Regalia, Rui Zhu, and Gengchen Mai. Debiasing knowledge graphs: Why female presidents are not like female popes. In *ISWC*, 2018.

[9] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In *Proc. Workshop on Representation Learning for NLP (RepL4NLP)*, pages 69–74, 2017.

[10] Jae Kwang Kim and Jay J. Kim. Nonresponse weighting adjustment using estimated response probability. *Canadian Journal of Statistics*, 35:501–514, 2007.

[11] Furong Li, Xin Luna Dong, Anno Langen, and Yang Li. Knowledge verification for long-tail verticals. *PVLDB*, 10:1370–1381, 2017.

[12] Hanxiao Liu, Yuexin Wu, and Yiming Yang. Analogical inference for multi-relational embeddings. In *ICML*, pages 2168–2178, 2017.

[13] Rushi Longadge and Snehlata Dongre. Class imbalance problem in data mining review. *International Journal of Computer Science and Network*, 2, 2013.

[14] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual Wikipedias. In *CIDR*, 2015.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[16] George A. Miller. WordNet: A lexical database for English. *Comm. ACM*, 38:39–41, 1995.

[17] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. In *Proc. IEEE*, 2016.

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

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

[20] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*, pages 6338–6347, 2017.

[21] Shameem P. Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing F-measures by cost-sensitive classification. In *NeurIPS*, pages 2123–2131, 2014.

[22] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8:489–508, 2017.

[23] Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *EMNLP*, 2017.

[24] Amit Singhal. Introducing the knowledge graph: Things, not strings, May 2012.

[25] Harald Steck. Training and testing of recommender systems on data missing not at random. In *SIGKDD*, pages 713–722, 2010.

[26] Harald Steck. Item popularity and recommendation accuracy. In *RecSys*, pages 125–132, 2011.

[27] Daria Stepanova, Mohamed H. Gad-Elrab, and Vinh Thinh Ho. Rule induction and reasoning over knowledge graphs. In *Reasoning Web International Summer School*, pages 142–172, 2018.

[28] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[29] Kristina Toutanova and Danqi Chen. Observed vs. latent features for knowledge base and text inference. In *Proc. Workshop on Continuous Vector Space Models and their Compositionality*, 2015.

[30] Théo Trouillon and Maximilian Nickel. Complex and holographic embeddings of knowledge graphs: a comparison. *arXiv preprint arXiv:1707.01475*, 2017.

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