Probabilistic Case-based Reasoning for Open-World Knowledge Graph Completion

Rajarshi Das, Ameya Godbole, Nicholas Monath, Manzil Zaheer, Andrew McCallum
University of Massachusetts, Amherst, USA
Google Research, USA
{rajarshi, agodbole, nmonath, mccallum}@cs.umass.edu
manzilzaheer@google.com

Abstract

A case-based reasoning (CBR) system solves a new problem by retrieving 'cases' that are similar to the given problem. If such a system can achieve high accuracy, it is appealing owing to its simplicity, interpretability, and scalability. In this paper, we demonstrate that such a system is achievable for reasoning in knowledge-bases (KBs). Our approach predicts attributes for an entity by gathering reasoning paths from similar entities in the KB. Our probabilistic model estimates the likelihood that a path is effective at answering a query about the given entity. The parameters of our model can be efficiently computed using simple path statistics and require no iterative optimization. Our model is non-parametric, growing dynamically as new entities and relations are added to the KB. On several benchmark datasets our approach significantly outperforms other rule learning approaches and performs comparably to state-of-the-art embedding-based approaches. Furthermore, we demonstrate the effectiveness of our model in an "open-world" setting where new entities arrive in an online fashion, significantly outperforming state-of-the-art approaches and nearly matching the best offline method.[1]

1 Introduction

We live in an evolving world with a lot of heterogeneity as well as new entities being created continuously. For example, scientific papers and Wikipedia pages describing facts about new entities, are being constantly added (e.g. COVID-19). These new findings further trigger the inference of newer facts, each with its own diverse reasoning. We are interested in developing such automated reasoning systems for large knowledge-bases (KBs).

In machine learning, non-parametric methods hold the promise of handling evolving data (Cover and Hart, 1967; Rasmussen, 2000). Most current KG completion models learn low dimensional parametric representation of entities and relations via tensor factorization or sophisticated neural approaches (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Sun et al., 2019; Vashishth et al., 2020). Another line of work learns Horn-clause style reasoning rules from the KG and stores them in its parameters (Rocktäschel and Riedel, 2017; Das et al., 2018; Minervini et al., 2020). However, these parametric approaches work with a fixed set of entities and it is unclear how these models will adapt to new entities.

This paper presents a k-nearest neighbor (KNN) based approach for KG reasoning that is reminiscent of case-based reasoning (CBR) in classical AI. A CBR system solves a new problem by retrieving 'cases' that are similar to the given problem, revising the solution to retrieved cases (if necessary) and reusing it for the new problem (Schank, 1982; Leake, 1996, inter-alia). For the task of finding a target entity given a source entity and binary KG relation (e.g. (JOHN VON NEUMAN, PLACE_OF_DEATH, ?) in Figure 1), our approach first retrieves k similar entities (cases) to the query entity. Next, for each retrieved entity, it finds multiple KG paths[2] (each path is a solution to retrieved cases) to the entity they are connected by the query relation (e.g. paths between (RICHARD FEYNMAN, USA)). However, one solution seldom works for all queries. For example, even though the path 'BORN_IN' is predictive of 'PLACE_OF_DEATH' for US-born scientists (figure 1), it does not work for scientists who have immigrated to USA. To handle this, we present a probabilistic CBR approach which learns to weighs paths with respect to an estimate of its prior and its precision, given the query. The prior of a path rep-

[1] Code available at https://github.com/ameyagodbole/Prob-CBR

[2] A path is a contiguous sequence of KG facts such as RICHARD FEYNMAN → AFFILIATED → CALTECH → LOCATED → USA.

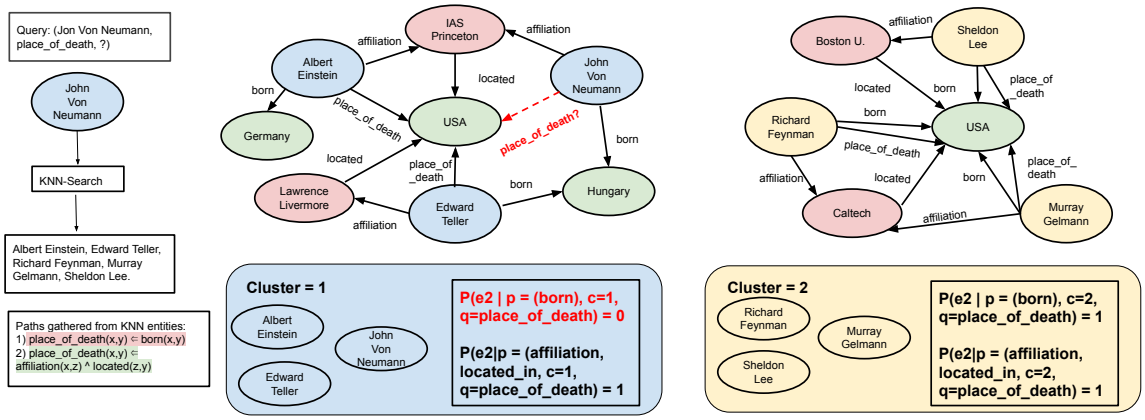

Figure 1: Given the query, (JON VON NEUMANN, PLACE_OF_DEATH, ?), our model gathers reasoning paths from similar entities such as other scientists. However, not all gathered paths work for a query e.g. the path ('BORN(x, y)') would not work for VON NEUMANN. This highlights the importance of learning path weights for *clusters of similar entities*. Even though 'BORN_IN' could be a reasonable path for predicting PLACE_OF_DEATH, this does not apply for VON NEUMANN and other scientists in his cluster. The precision parameter of the path given the cluster helps in penalizing the 'BORN_IN' path. Note that the node USA is repeated twice in the figure to reduce clutter.

resents its frequency while the precision represents the likelihood that the path will lead to a correct answer entity. To obtain robust estimates of the path parameters, we cluster similar entities together and compute them by simple count statistics (§2.2.3).

Apart from computing these estimates, our method needs *no further training*. Overall, our simple approach outperforms several recent parametric rule learning methods (Das et al., 2018; Minervini et al., 2020) and performs competitively with various state-of-the-art KG completion approaches (Dettmers et al., 2018) on multiple datasets.

An advantage of non-parametric models is that it can adapt to growing data by adjusting its number of parameters. In the same spirit, we show that our model can seamlessly handle an 'open-world' setting in which *new entities* arrive in the KG. This is made possible by several design choices such as (a) representing entities as sparse (non-learned) vector of its relation types (§2.2.1), (b) our use of an online non-parametric hierarchical clustering algorithm (Monath et al., 2019) that can efficiently recompute changes in cluster assignments because of the newly added entity (§2.3), (c) and a simple and efficient way of recomputing the prior and precision parameters for paths per cluster (§2.2.3).

Current models for KG completion that learn entity representations for a fixed set of entities cannot handle the open-world setting. In fact we show that, retraining the models continually with new data leads to severe degradation of the model performance with models forgetting what it had learned before. For example, the performance (MRR) of ROTATE model (Sun et al., 2019) drops by 11 points (absolute) on WN18RR in this setting (§3.4). On the other hand, we show that with new data, the performance of our model is consistent as it is able to seamlessly reason with the newly arrived data.

Our work is most closely related to a recent concurrent work by Das et al. (2020) where they propose a model that gathers paths from entities similar to the query entity. However, Das et al. (2020) encourages path that occur frequently in the KG and does not learn to weigh paths differently for queries. This often leads to wrong inference leading to low performance. For example, on the test-II evaluation subset of FB122 where all triples can be inferred by logical rules, Das et al. (2020) scores quite low (63 MRR) because of learning incorrect rules. On the other hand, we score significantly higher (94.83 MRR) demonstrating that we can learn more effective rules. In fact, we consistently and significantly outperform Das et al. (2020) on several benchmark datasets. Also, unlike us, they do not test themselves in the challenging open-world setting.

The contributions of this paper are as follows: (a) We present a KNN based approach for KG completion that gathers reasoning paths from entities that are similar to the query entity. Following a principled probabilistic approach (§2.2), our model weighs each path by its likelihood of reaching a correct answer which penalizes paths that are spurious in nature. (b) The parameters of our model grow with data and can be estimated efficiently using simple count statistics (§2.3). Apart from this, our approach needs *no training*. We show that our

simple approach significantly outperforms various rule learning methods (Das et al., 2018; Minervini et al., 2020; Das et al., 2020) on many benchmark datasets. (c) We also show that our model can easily handle addition of facts about new entities and is able to seamlessly integrate and reason with the newly added data significantly outperforming parametric embedding based models.

2 Non-parametric Reasoning in KGs

2.1 Notation and Task Description

Let $\mathcal{V}$ denote the set of entities, $\mathcal{R}$ denote the set of binary relations and $\mathcal{G}$ denote a KB or equivalently a Knowledge Graph (KG). Formally, $\mathcal{G} = (\mathcal{V}, E, \mathcal{R})$ is a directed labeled multigraph where $\mathcal{V}$ and E denote the vertices and edges of the graph respectively. Note that, $E \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$. Let (e_1, r, e_2) denote a fact in $\mathcal{G}$ where $e_1, e_2 \in V$ and $r \in E$. Also, following previous approaches (Bordes et al., 2013), we add the inverse relation of every edge, i.e., for an fact $(e_1, r, e_2) \in E$, we add the edge (e_2, r^{-1}, e_1) to the graph. (If the set of binary relations $\mathcal{R}$ does not contain the inverse relation r^{-1}, it is added to $\mathcal{R}$ as well).

Task: We consider the task of query answering on KGs, i.e., answering questions of the form $(e_{1q}, r_q, ?)$, where answer is an entity in the KG.

Paths in KG: A path in a KG between two entities e_s, e_t is defined as a sequence of alternating entity and relations that connect e_s and e_t. A length of a path is the number of relation (edges) in the path. Formally, let a path $p = (e_1, r_1, e_2, \ldots, r_n, e_{n+1})$ with $\mathsf{st}(p) = e_1$, $\mathsf{en}(p) = e_{n+1}$ and $\mathsf{len}(p) = n$. We also define a *path type* as the sequence of the relations in p, i.e., $\mathsf{type}(p) = (r_1, r_2, \ldots, r_n)$. Let $\mathcal{P}$ denote the set of all paths in $\mathcal{G}$. Let $\mathcal{P}_n \subseteq \mathcal{P} = \{p \mid \mathsf{len}(p) \leq n\}$ be the set of all paths of length up to n. Also, let $\mathrm{P_n}$ denote the set of all path *types* with length up to n, i.e. $\mathrm{P_n} = \{\mathsf{type}(p) \mid p \in \mathcal{P}_n\}$. Let $\mathrm{P_n}(e_1, r) \subseteq \mathrm{P_n}$ denote all path types of length up to n that originate at e_1 and end at the entities that are connected to e_1 by a direct edge of type r. In other words, if $S_{e_1 r} = \{e_2 \mid (e_1, r, e_2) \in \mathcal{G}\}$ denotes the set of entities that are connected to e_1 via a direct edge r, then $\mathrm{P_n}(e_1, r)$ denotes the set of all path types of length up to n that start from e_1 and end at entities in $S_{e_1 r}$. By definition, $r \in \mathrm{P_n}(e_1, r)$. Similarly, we define $\mathcal{P}_n(e_1, r)$ which contain paths instead of path types.

2.2 Model

Given a query, our approach gathers KG path types from entities that are similar to the query entity. Each path type is weighed with respect to an estimate of both its frequency and precision (§2.2.1). By clustering similar entities together (§2.2.2), our model obtains robust estimate of the path statistics (§2.2.3). Our approach is non-parametric because - (a) Instead of storing reasoning rules in parameters (Das et al., 2018; Minervini et al., 2020), it derives them dynamically from k-similar entities (like a non-parametric k-nn classifier (Cover and Hart, 1967)). (b) We cluster entities together using a non-parametric clustering approach and provide an efficient way of adding / estimating parameters when entities are added to the KG (§2.3).

2.2.1 Reasoning from contextual entities

Our approach first finds k similar entities to the query entity that have atleast an edge of type r_q. For example, for the query (MELINDA GATES, WORKS_IN_CITY, ?), we would consider WARREN BUFFET if we observe (WARREN BUFFET, WORKS_IN_CITY, OMAHA). We refer to these entities as 'contextual entities'. Each entity is represented as a sparse vector of its outgoing edge types, i.e. $\mathbf{e_i} \in \{0,1\}^{|\mathcal{R}|}$. If entity e_i has m distinct outgoing edge types, then the dimension corresponding to those types are set to 1. This is an extremely simple and flexible way of representing entities which we find to work well. Also note that, as more data is added about an entity, this sparse representation makes it trivial to update the embeddings.

Let $E_{c,q}$ denote the set of contextual entities for the query q. To compute $E_{c,q}$, we first sort entities with respect to their cosine distance with respect to query entity and select the k entities with the least distance and which have the query relation r_q. For each contextual entity e_c, we gather the path types (up to length n) that connect e_c to the entities it is connected by the edge r_q (i.e. $\mathrm{P_n}(e_c, r_q)$ in §2.1). These extracted path types will be used to reason about the query entity. Let $\mathrm{P_n}(E_{c,q}, r_q) = \bigcup_{e_c \in E_{c,q}} \mathrm{P_n}(e_c, r_q)$ represent the set of unique path types from the contextual entities. The probability of finding the answer entity e_2 given the query is given by:

$$
\begin{aligned}
P(e_2 \mid e_{1q}, r_q) &= \sum_{p \in \mathrm{P_n}(E_{(c,q)}, r_q)} P(e_2, p \mid e_{1q}, r_q) \\
&= \sum_p P(p \mid e_{1q}, r_q) P(e_2 \mid p, e_{1q}, r_q) \quad (1)
\end{aligned}
$$

We marginalize the random variable representing the path types obtained from $E_{c,q}$. $P(p \mid e_{1q}, r_q)$ denotes the probability of finding a path type given the query. This term captures how frequently each path type co-occurs with a query and represents the prior probability for a path type. On the other hand, $P(e_2 \mid p, e_{1q}, r_q)$ captures the proportion of times, when a path type p is traversed starting from the query entity, we reach the correct answer instead of some other entity. This term can be understood as capturing the likelihood of reaching the right answer or the 'precision' of a reasoning path type. This is crucial in penalizing 'spurious' path types that sometimes coincidentally find the right answer entity. For example, for the query relation WORKS_IN_CITY, the path type (FRIEND∧ LIVES_IN_CITY) might have a high prior probability (since people often have many friends in the city where they work). However, this path is 'spurious' with respect to WORKS_IN_CITY, since they might have friends living in various cities and hence this path type will not necessarily return the correct answer.

2.2.2 Entity Clustering

Equation 1 has parameters for each entity in the KG. For large KGs, this can quickly lead to parameter explosion. Also, estimating per-entity parameter leads to noisy estimates due to sparsity. Instead, we choose to cluster similar entities together. Let c be a random variable representing the cluster assignment of the query entity. Then for the path-prior term, we have

$$P(p \mid e_{1q}, r_q) = \sum_c P(c \mid e_{1q}, r_q) P(p \mid c, e_{1q}, r_q)$$

We assume that each entity is assigned to one cluster, so $P(c \mid e_{1q}, r_q)$ is zero for all clusters except the cluster in which the query entity belongs to. Secondly we assume, that the prior probability of a path given the entity and cluster can be determined from the cluster alone and is independent of each entity in the cluster. In other words, if $c_{e_{1q}}$ is the cluster in which the $e_{1,q}$ has been assigned, then $P(p \mid c_{e_{1q}}, e_{1q}, r_q) = P(p \mid c_{e_{1q}}, r_q)$. Instead of per-entity parameters, we now aggregate statistics over entities in the same cluster and have per-cluster parameters. We also show that this leads to significantly better performance (§3.3). A similar argument applies for the path-precision term in which we calculate the proportion of times, a path leads to the correct answer entity starting from each entity in the cluster.

To perform clustering, we use hierarchical agglomerative clustering with average linkage with the entity-entity similarity defined in §2.2.1. We extract a non-parameteric number of clusters from the hierarchy using a threshold on the linkage function. Agglomerative clustering has been shown to be effective in many knowledge-base related tasks such as entity resolution (Lee et al., 2012; Vashishth et al., 2018) and in general has shown to outperform flat clustering methods such as K-means (Green et al., 2012; Kobren et al., 2017). A flat clustering is extracted from the hierarchical clustering by using a threshold on the linkage function score. We perform a breadth first search from the root of the tree stopping at nodes for which the linkage is above the given threshold. The nodes where the search stops give a flat clustering (refer to §A.2 for more detail on this).

2.2.3 Parameter Estimation

Next we discuss how to estimate path prior and precision terms. There exists abundant modeling choices to estimate them. For example, following Chen et al. (2018), we could train a neural network model to estimate $P(p \mid c_{e_{1q}}, r_q)$. However, with our original goal of designing a simple and efficient non-parametric model, we estimate these parameters by simple count statistics from the KG. E.g., the path prior $P(p \mid c, r_q)$ is estimated as

$$\frac{\sum_{e_c \in c} \sum_{p' \in \mathcal{P}_n(e_c, r_q)} \mathbb{1}\left[\mathsf{type}(p') = p\right]}{\sum_{e_c \in c} \sum_{p' \in \mathcal{P}_n(e_c, r_q)} \mathbb{1}} \tag{2}$$

For each entity in cluster c, we consider the paths that connect e_c to entities it is directly connected to via edge type r_q ($\mathcal{P}_n(e_c, r_q)$ in §2.1). The path prior for a path type p is computed as the proportion of times the type of paths in $\mathcal{P}_n(e_c, r_q)$ is equal to p. Note that in equation 2, if a path type appears multiple times, we count all instances. For example, for the query relation WORKS_IN_CITY, a path of the form (CO_WORKER ∧ WORKS_IN_CITY) can occur multiple times, since a person can have multiple different co-workers. Considering just path types will lead to under-weighing of such important paths. Similarly, the path-precision probability ($P(e_2 \mid p, c, r_q)$) can be estimated as,

$$\frac{\sum_{e_c \in c} \sum_{p' \in \mathcal{P}_n(e_c)} \mathbb{1}[\mathsf{type}(p') = p] \cdot \mathbb{1}[\mathsf{en}(p') \in S_{e_c r_q}]}{\sum_{e_c \in c} \sum_{p' \in \mathcal{P}_n(e_c)} \mathbb{1}[\mathsf{type}(p') = p]} \tag{3}$$

Let $\mathcal{P}_n(e_c)$ denote the paths of up to length n starting from the entity e_c. Note, unlike $\mathcal{P}_n(e_c, r_q)$, the

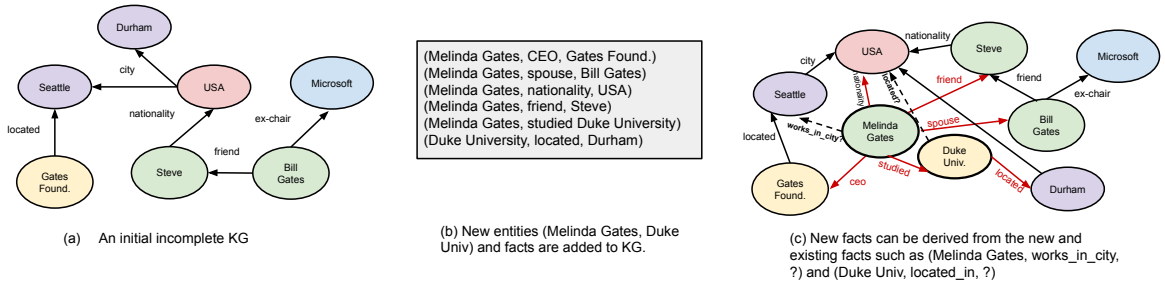

Figure 2: We consider a setting where *new entities* and facts are added continuously to the KG. Our non-parametric approach can seamlessly reason with the newly added entities and can infer new facts about them (e.g. (MELINDA, WORKS_IN_CITY, ?) or (DUKE UNIV., LOCATED_IN_COUNTRY, ?)) without requiring expensive training.

paths in $\mathcal{P}_n(e_c)$ do not have to end at specific entities. Also from §2.1, $\text{en}(p)$ denotes the end entity for a path p and $S_{e_c r_q}$ denotes the set of entities that are connected to e_c via a direct edge of type r_q. Equation 3, therefore, estimates the proportion of times the path p successfully ends at one of the answer entities when starting from e_c, given r_q.

There are several advantages in estimating the parameters using simple count statistics. Firstly, they are extremely simple, and statistics for each entity in clusters can be computed in parallel making them extremely time efficient. Secondly once they are computed, our approach needs *no further training*. Lastly, when new data is added, it makes it easy to update the parameters without training from scratch.

To summarize, given a query entity (e_{1q}, r_q), our method gathers reasoning paths from k similar entities to e_{1q}. These reasoning paths are then traversed in the KG starting from e_{1q}, leading to a set of candidate answer entities. The score of each answer entity candidate is computed as a weighted sum of the reasoning paths the lead to them (Equation 1). Each path is weighed with an estimate of its frequency (Equation 2) and precision (Equation 3) given the query relation. The next section describes how we extend our model for open-world setting where new entities and facts are added to the KB.

2.3 Open-world Setting

A great benefit of non-parametric models is that it can seamlessly handle growing data by adding new parameters. New entities constantly arrive in the world (e.g. new Wikipedia articles about entities are frequently created). We consider a setting (Figure 2) in which new entities with few facts (edges) about them keep getting added to the KG. This setting is challenging for parametric models (Das et al., 2018; Sun et al., 2019) as it is unclear how these models can incorporate new entities without retraining from scratch. However, retraining to obtain entity embeddings on industrial scale KGs might be impractical (e.g. consider Facebook social graph where new users are joining continuously). Next, we show that our approach can handle this setting efficiently in the following way:

(a) **Adding/updating entity representations**: First we need to create entity representations for the newly arrived entities. Also, for some existing entities for which new edges were added (e.g. BILL GATES, DURHAM, etc. in figure 2), their representations need to be updated. Recall, that we represent entities as a sparse vector of its edge types and hence this step is trivial for our approach.

(b) **Updating cluster assignments**: Next the new entities needs to be added to clusters of similar entities. Also, the cluster assingments of entities that got updated can also change as well and their change can further trigger changes to the clustering of other entities. To handle this, one could naively cluster all entities in the KG, however that could be wasteful and time-consuming for large KGs. Instead, we use an online hierarchical clustering algorithm - GRINCH (Monath et al., 2019), which has shown to perform as well as agglomerative clustering in the online setting. GRINCH observes one entity at a time, placing it next to its nearest neighbor and performing local re-arrangements in the form of rotations of tree nodes and global re-arrangments in the form of grafting a subtrees from part of the tree to another. Entities can be deleted from a hierarchy by simply removing the corresponding leaf node. We first use GRINCH to delete the entities whose representations had changed because of the addition of the new node and then incrementally add those entities back along with the newly added entities in the KG. We extract a flat clustering from the hierarchical clustering built

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert E\rvert$
NELL-995	75,492	200	154,213
FB122	9,738	122	112,476
WN18RR	40,943	11	93,003

Table 1: Dataset Statistics

by GRINCH using the same method as in §2.2.2.
(c) **Re-estimating new parameters**: After re-assigning clusters, the final step is to estimate the per-cluster parameters. This computation is efficient as it is clear from equations 2 and 3 that the contribution from each entity in a cluster can be computed independently (and hence can be easily parallelized). However, even for each entity, this computation needs path traversal in the KG which is expensive. We show that we do not have to re-compute for all entities in the clusters.

Let n denote the maximum length of a reasoning path considered by our model. For every new entity e_i added to the KG, we need to recompute statistics for entities that lie within cycles of length up to $(n+1)$ starting from e_i. Please refer to appendix (A.4) for a justification of this result.

3 Experiments

In this section, we evaluate our proposed approach on a wide array of knowledge-base completion (KBC) benchmarks (§3.3). To evaluate the non-parametric nature of our approach, we also evaluate on an 'open-world' setting (§2.3) in which new entities are added to the KG. We demonstrate our proposed approach is competitive to several state-of-the-art methods on benchmarks in the standard setting, but it greatly outperforms other methods in the online setting (§3.4). The best hyper-parameters for all experiments including the range of hyper-parameter tried and results on validation set are noted in §A.6.

3.1 Data and Evaluation Protocol

Data. We evaluate on the following KBC datasets: **NELL-995**, **FB122** (Guo et al., 2016), **WN18RR** (Dettmers et al., 2018). **FB122** is a subset of the dataset derived from Freebase, FB15K (Bordes et al., 2013), containing 122 relations regarding people, locations, and sports. NELL-995 (Xiong et al., 2017) a subset of the NELL derived from the 995th iteration of the system. WN18RR was created by Dettmers et al. (2018) from WN18 by removing inverse relation test-leakage.

Evaluation metrics. Following previous work, we evaluate our method using HITS@N and mean reciprocal rank (MRR), which are standard metrics for evaluating a ranked list.

3.2 Experimental Setting

Knowledge Base Completion. Given an entity e_1 and a relation r, our task is retrieve all entities e_2 such that (e_1, r, e_2) belongs in the edges E in a KG $\mathcal{G}$. This task is known as *tail prediction*. If the relation is instead the inverse relation r^{-1}, we assume that we are given an e_2' and asked to predict entities e_1' such that (e_1', r^{-1}, e_2') belongs in the edges E (*head prediction*). To be exactly comparable to baselines, we report an average of head and tail prediction results[3]. We are given a knowledge graph with three partitions of edges, E_{train}, E_{dev}, E_{test}.

For this task, we evaluate against several state-of-the-art embeddings based models such as DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), RotatE (Sun et al., 2019). We also compare against several parametric rule learning methods — NTP (Rocktäschel and Riedel, 2017), NeuralLP (Yang et al., 2017), MINERVA (Das et al., 2018), GNTP (Minervini et al., 2020) and also the closely related CBR approach of Das et al. (2020).

Open-world Knowledge Base Completion. In this setting, we begin with the top 10% of the most popular nodes (with several edges going out from them) and add more randomly selected nodes such that the initial seed KB contains 50% of all the entities in $\mathcal{V}$. This is to ensure, that the seed KB is not too sparse and the initial models trained on them are meaningful. Next, any edges between the nodes selected are added to the seed KB. We divide the rest of the entities randomly into 10 batches. Each batch of entities is incrementally added to the KB along with the edges contained in it. The validation and test set are also divided in the same way, i.e. if both the head and tail entity of a triple are present in the KB, only then the triple is put in the corresponding splits.

Parametric models for KBC that learn representations for a fixed set of entities can not handle 'open-world' setting out-of-the-box. We extend the most competitive embedding based model - RotatE (Sun et al., 2019) for this task. For every new entity arriving in a batch, we initialize a new entity embedding for it. We explore two ways of initial-

[3]except for NELL-995 dataset where like our baselines, we report tail-prediction performance.

		Test-I				Test-II				Test-ALL			
		Hits@N (%)			MRR	Hits@N (%)			MRR	Hits@N (%)			MRR
		3	5	10		3	5	10		3	5	10	
With Rules	KALE-Pre (Guo et al., 2016)	35.8	41.9	49.8	0.291	82.9	86.1	89.9	0.713	61.7	66.2	71.8	0.523
	KALE-Joint (Guo et al., 2016)	38.4	44.7	52.2	0.325	79.7	84.1	89.6	0.684	61.2	66.4	72.8	0.523
	ASR-DistMult (Minervini et al., 2017)	36.3	40.3	44.9	0.330	98.0	99.0	99.2	0.948	70.7	73.1	75.2	0.675
	ASR-ComplEx (Minervini et al., 2017)	37.3	41.0	45.9	0.338	**99.2**	**99.3**	**99.4**	**0.984**	71.7	73.6	75.7	0.698
	KBLR (Garcia-Duran and Niepert, 2018)	–	–	–	–	–	–	–	–	74.0	**77.0**	**79.7**	0.702
Without Rules	TransE (Bordes et al., 2013)	36.0	41.5	48.1	0.296	77.5	82.8	88.4	0.630	58.9	64.2	70.2	0.480
	DistMult (Yang et al., 2015)	36.0	40.3	45.3	0.313	92.3	93.8	94.7	0.874	67.4	70.2	72.9	0.628
	ComplEx (Trouillon et al., 2016)	37.0	41.3	46.2	0.329	91.4	91.9	92.4	0.887	67.3	69.5	71.9	0.641
	GNTPs (Minervini et al., 2020)	33.7	36.9	41.2	0.313	**98.2**	**99.0**	**99.3**	**0.977**	69.2	71.1	73.2	0.678
	RotatE (Sun et al., 2019)	**51.1**	**55.1**	**60.3**	**0.471**	86.8	88.6	90.7	0.846	70.8	73.57	77.0	0.678
	CBR (Das et al., 2020)	40.0	44.5	48.8	0.359	67.8	71.8	75.9	0.636	57.0	61.2	65.3	0.527
	Our Model	49.0	52.7	57.1	0.457	94.8	95.0	95.3	0.948	**74.2**	**76.0**	**78.2**	**0.727**

Table 2: Link prediction results on FB122. Test-II denotes a subset of triples that can be inferred via logical rules.

Metric	TransE	DistMult	ComplEx	ConvE	RotatE	GNTP	MINERVA	CBR	Our Model
HITS@1	-	0.39	0.41	0.40	**0.43**	0.41	0.40	0.38	**0.43**
HITS@3	-	0.44	0.46	0.44	**0.49**	0.44	0.43	0.46	**0.49**
HITS@10	0.50	0.49	0.51	0.52	**0.57**	0.48	0.49	0.51	0.55
MRR	0.23	0.43	0.44	0.43	**0.48**	0.43	0.43	0.43	**0.48**
HITS@1	0.53	0.61	0.61	0.67	0.65	-	0.66	0.70	**0.77**
HITS@3	0.79	0.73	0.76	0.81	0.82	-	0.77	0.83	**0.85**
HITS@10	0.87	0.79	0.83	0.86	0.87	-	0.83	0.87	**0.89**
MRR	0.67	0.68	0.69	0.75	0.74	-	0.72	0.77	**0.81**

Table 3: Results on WN18RR (above) and NELL-995 (tail-prediction;below)

izing the new entity embeddings — (a) random initialization, and (b) average of element-wise rotation of entity embeddings w.r.t the relation that this new entity is connected to. Specifically, let t denote the new entity and let $S = \{(h,r,t)\}$ be the facts associated with entity t. Then the embedding $\mathbf{e_t}$ is computed as

$$\mathbf{e_t} = \frac{\sum_{(h,r,t)\in S} \mathbf{e_h} \circ \mathbf{e_r}}{|S|} \tag{4}$$

Here, $\circ$ represents the Hadamard (or element-wise) product. This initialization minimizes the RotatE objective for the new embedding ensuring that it is "well-placed" according to the model in the previous time step. Embeddings for new relations are initialized randomly. Next, the model is further trained on the new batch of triples so that the new entity embeddings get trained. Note, for massive KGs, it might be impractical to re-train on the entire data as new batches of data arrive frequently, however to still prevent the model to forget what it had learned before, we also sample $m\%$ of triples that it had already been trained on and re-train on them. We ensure that triples in the neighborhood of the newly added entities are ten times likely to be sampled more than other triples. We also try a setting where we try freezing the initially trained entity embeddings and only training the new entity and relation embeddings.

3.3 Results on KBC benchmarks

The results for KBC tasks are presented in Table 2 and 3[4]. Our method does significantly better than parametric rule learning approaches such as MINERVA, GNTPs and the recent case-based approach of Das et al. (2020). We would like to highlight the difference between the performance of our model and that of Das et al. (2020) on the test-II evaluation of FB122 where triples can be answered by learning logical rules. This results emphasizes the importance of our probabilistic weighing of paths. We also perform comparably to most embedding based models and achieve state-of-the-art results on the overall test sets of FB122 and NELL-995. We report the mean over 3 runs for our model.

We perform an ablation where we do not cluster entities (i.e. every entity has its own cluster) and have per-entity parameters. Table 4 notes the drop in performance due to the noisy estimates of path prior and precision parameters because of sparsity. Table 6 shows an example where our model learns to score different paths based on the type of entities present in the cluster.

Effect of path length on WN18RR: On the dev set of WN18RR, out of 2985 queries where

[4]There are no reported results of GNTPs on NELL-995

	Our Method	Our Method w/o clustering
HITS@1	0.42	0.29
HITS@3	0.46	0.36
HITS@10	0.51	0.45
MRR	0.45	0.34

Table 4: Impact of clustering on WN18RR

	RotatE	Our Method ($n = 3$)	Our Method ($n = 5$)
HITS@1	0.43	0.42	0.43
HITS@3	0.49	0.46	0.49
HITS@10	0.57	0.51	0.55
MRR	0.48	0.45	0.48

Table 5: Impact of path length on WN18RR

our method does not rank the answer in the top-10, 2030 queries require a minimum path length greater than 3. Path-based reasoning models have no power to answer these queries. To correct for this, we perform an experiment with the path length $n = 5$ (950 of 2030 answers are reachable). The results in Table 5 show that our method recovers a significant portion of performance when allowed to use longer reasoning paths.

3.4 Open-World KBC results

Figure 3 reports the result for this task. We report results on the RotatE model with randomly initialized embeddings for new entities (RotatE) and the model with systematic initialization of new entity embeddings (RotatE+). We experiment with $m = \{10\%, 30\%\}$ of previously seen edges and retrain on them. We find that not including previously seen edges leads to severe degradation of overall performance due to the model forgetting what it had learned in the past. We also report results with freezing the already seen entity representations and only learning representations for new entities (RotatE-Freeze). All models were trained till the validation set (containing both new and old triples) performance stopped improving. For our approach, we also report results for an oracle setting where we re-cluster all entities as new data arrives and re-estimate all parameters from scratch (instead of using GRINCH and recomputing only required parameters (§2.3). For both datasets, the offline-best results were obtained by RotatE (47.1 for FB122 test-I, 48 for WN18RR). We report performance on the entire evaluation set (full) and also on the set containing the newly added edges (new).

The main summary of the results are (i) RotatE model converges to a much lower performance in the online setting losing at least 8 MRR points in FB122 and at least 11 points in WN18RR. On FB122, we observe that the model prefers to learn new information more by sacrificing previously learned facts (2nd subfigure in figure 3) (ii) In the freeze setting, the model performance deteriorates quickly after a certain point indicating saturation, i.e. it becomes hard for the model to learn new information about arriving entities by keeping the parameters of the existing entities fixed. (iii) On the full evaluation, RotatE+ performs better than RotatE showing that bad initialization deteriorates performance over time, however, there is still a large gap between the best performance (iv) Our approach almost matches our performance in oracle setting indicating the effectiveness of the online clustering and fast parameter approximation. (v) Lastly, we perform closest to the offline best results outperforming all variants of RotatE.

4 Related Work

Open-world KG completion. Shi and Weninger (2018) consider the task of open-world KG completion. However, they use text descriptions to learn entity representations using convolutional neural networks. Our model does not use additional text data and we use very simple entity representations that helps us to perform well. Tang et al. (2019) learns to update a KG with new links by reading news. Even though they handle adding or deleting new edges, they do not observe new entities. Lastly, none of them learn from similar entities using a CBR approach.

Inductive representation learning on KGs. Recent works (Teru et al., 2020; Wang et al., 2020) learn entity independent relation representations and hence allow them to handle unseen entities. However, they do not perform contextual reasoning by gathering reasoning paths from similar entities. Moreoever, in our open-world setting, we consider the more challenging setting, where new facts and entities are arriving in a streaming fashion and we give an efficient way of updating parameters using online hierarchical clustering. This allows our method to be applicable in settings where the initial KG is small and it grows continuously.

Rule induction in knowledge graphs. Classic work in inductive logic programming (ILP) (Muggleton et al., 1992; Quinlan, 1990) induce rules from grounded facts. However, they need explicit counter-examples which are not present in KBs and they do not scale to large KBs. Recent ILP approaches (Galárraga et al., 2013, 2015) try to fix

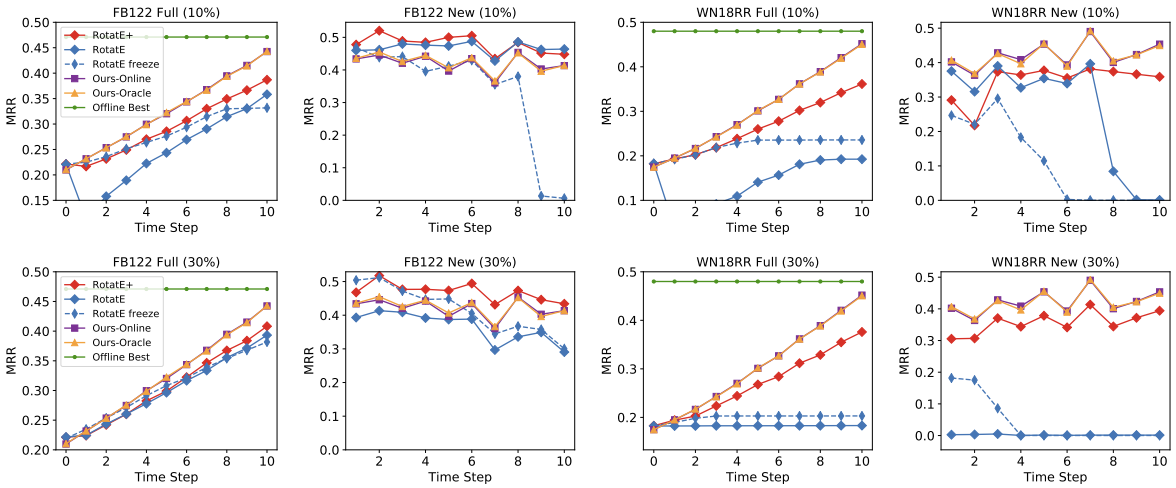

Figure 3: Results for open-world setting when trained with 10% (top row) and 30% (bottom row) of already seen edges. Our online method matches the offline version of our approach and outperforms the online variants of RotatE. After all data is observed our online method achieves results closest to the best offline method's results.

Athlete Cluster	(athlete-led-sports-team, team-plays-in-league) (athlete-home-stadium, league-stadiums^{-1})
Politician Cluster	(politician-us-member-of-political-group, person-belongs-to-organization^{-1}, agent-belongs-to-organization) (agent-collaborates-with-agent, agent-belongs-to-organization)

Table 6: High scoring paths in different clusters for the query agent-belongs-to-organization in NELL-995

this deficiency by guessing counter examples from rules and making it more scalable. Statistical relational learning methods (Getoor and Taskar, 2007; Kok and Domingos, 2007; Schoenmackers et al., 2010) and probabilistic logic approaches (Richardson and Domingos, 2006; Broecheler et al., 2010; Wang et al., 2013) combine machine learning and logic to learn rules. However, none of these work derive reasoning rules dynamically from similar entities in the knowledge graph.

Bayesian non-parametric approaches for link-prediction. There is a rich body of work in bayesian non-parametrics to automatically learn the latent dimension of entities (Kemp et al., 2006; Xu et al., 2006). Our method does not learn latent dimension of entities, instead our work is non-parametric because it gathers reasoning paths from nearest neighbors and can seamlessly reason with new entities by efficiently updating parameters using online non-parametric hierarchical clustering.

Embedding-based approach for link prediction. We also compare to the more popular embeddings based models based on tensor factorization or neural approaches (Nickel et al., 2011; Bordes et al., 2013; Dettmers et al., 2018; Sun et al., 2019). Our simple approach which needs no iterative optimization outperforms most of them and performs comparably to the latest RotatE model. Moreover we outperform RotatE in the online experiments.

CBR for KG completion. There has been few attempts to apply CBR for knowledge management (Dubitzky et al., 1999; Bartlmae and Riemenschneider, 2000), however they do not do contextualized reasoning or consider online settings. Our work is most closely related to the recent work of Das et al. (2020). However, since it does not take in to account the importance of each path, it suffers from low performance, with our model outperforming it in several benchmarks.

5 Conclusion

We present a simple yet accurate approach for probabilistic case-based reasoning in knowledge bases. Our method is non-parametric, deriving reasoning rules dynamically from similar entities in the KB and is capable of handling new entities. We cluster similar entities together and estimate per-cluster parameters that measures the prior and precision of paths using simple count statistics. Our simple approach performs competitively to the best embeddings based models on several benchmarks and outperforms all models in the open-world setting.

Acknowledgements

We thank anonymous reviewers and members of UMass IESL and NLP groups for helpful discussion and feedback. This work is funded in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part by the National Science Foundation under Grants No. IIS-1514053 and No. 1763618, and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Kai Bartlmae and Michael Riemenschneider. 2000. Case based reasoning for knowledge management in kdd projects. In *PAKM*.

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

Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic similarity logic. In *UAI*.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. In *NAACL*.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*.

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*.

Rajarshi Das, Ameya Gobole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2020. Non-parametric reasoning in knowledge bases. In *AKBC*.

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

W Dubitzky, AG Büchner, and FJ Azuaje. 1999. Viewing knowledge management as a case-based reasoning application. In *AAAI Workshop Technical Report*, pages 23–27.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. 2015. Fast rule mining in ontological knowledge bases with amie+. In *VLDB*.

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 and Mathias Niepert. 2018. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *UAI*.

Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. MIT press.

Spence Green, Nicholas Andrews, Matthew R Gormley, Mark Dredze, and Christopher D Manning. 2012. Entity clustering across languages. In *NAACL-HLT*.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *EMNLP*.

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*.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. 2017. A hierarchical algorithm for extreme clustering. In *KDD*.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *ICML*.

David B Leake. 1996. Cbr in context: The present and future. *Case-based reasoning: Experiences, lessons, and future directions*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *EMNLP/CoNLL*.

Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. 2020. Differentiable reasoning on large knowledge bases and natural language. In *AAAI*.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*.

Nicholas Monath, Ari Kobren, Akshay Krishnamurthy, Michael R Glass, and Andrew McCallum. 2019. Scalable hierarchical clustering with tree grafting. In *KDD*.

Stephen Muggleton, Ramon Otero, and Alireza Tamaddoni-Nezhad. 1992. *Inductive logic programming*. Springer.

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

J Ross Quinlan. 1990. Learning logical definitions from relations. *Machine learning*.

Carl Edward Rasmussen. 2000. The infinite gaussian mixture model. In *Neurips*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *NeurIPS*.

Roger C Schank. 1982. *Dynamic memory: A theory of reminding and learning in computers and people*. cambridge university press.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *EMNLP*.

Baoxu Shi and Tim Weninger. 2018. Open-world 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*.

Jizhi Tang, Yansong Feng, and Dongyan Zhao. 2019. Learning to update knowledge graphs by reading news. In *EMNLP*.

Komal K Teru, Etienne Denis, and William L Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *ICML*.

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

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *WWW*.

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

Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2020. Entity context and relational paths for knowledge graph completion. *arXiv preprint arXiv:2002.06757*.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *CIKM*.

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

Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. 2006. Infinite hidden relational models. In *UAI*.

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*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NeurIPS*.

Algorithm 1 Select a flat clustering from a tree structure.

input: $\mathcal{V}$: Entities , *root*: Root of tree, τ: Threshold
output: $C_1, C_2, \ldots, C_K$: A flat partition
$frontier \leftarrow [root]$
$result \leftarrow \{\}$
while $frontier$ is not empty **do**
  $n \leftarrow frontier.pop()$
  if $linkage(n) > \tau$ **then**
    $result \leftarrow \{n\} \cup result$
  else
    for c in $n.children$ **do**
      $frontier.push(c)$
    end for
  end if
end while
return *result*

A Appendix

A.1 Entity Clusters

Both clustering methods used in this paper, hierarchical agglomerative clustering (HAC) and GRINCH measure similarities between sets of clusters via a linkage function. In particular, we use average pairwise linkage. For two sets A and B, this is defined as:

$$\frac{1}{|A||B|} \sum_{a \in A} \sum_{a \in B} \mathsf{sim}(a, b) \tag{5}$$

A.2 Selecting Flat Clusterings

A hierarchical clustering T over the entities $\mathcal{V}$, encodes a large number of flat partitions of the entities, often referred to as tree consistent partitions in the clustering literature. We select one of these tree consistent partitions using a threshold on the linkage function, τ. The algorithm performs a breadth first search starting at the root node. The search stops at any node for which the linkage is above the given value τ. Pseudocode is given in Algorithm 1.

A.3 Number of Entity Updates Per Batch In Online Setting

We analyze the number of entities that need to be re-clustered and added in each round. We observe that it is significantly fewer than the number of entities in the KB. Note that an online method like the one proposed in this paper just needs to run on the new and modified entities while a batch algorithm would need to run on the entire KB.

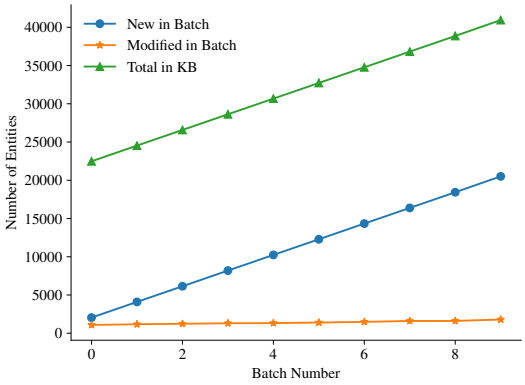

Figure 4: Number of entities added to KB in each batch and number of entities modified in each batch. These new and modified entities need to be updated in the clustering algorithm in each update.

A.4 Finding entities for re-estimating parameters

Proposition: Let n denote the maximum length of a reasoning path considered by our model. For every new entity e_i added to the KG, we need to recompute statistics for entities that lie within cycles of length up to $(n+1)$ starting from e_i.

We see from Eq 2, that the estimate for the prior for a *path type* p depends on $\mathcal{P}_n(e_c, r_q)$ i.e. the set of paths that lead from e_c to entities that are connected to e_c via relation r_q. WLOG, say e_t is such an entity i.e. $(e_c, r_q, e_t) \in \mathcal{G}$. When a new entity/edge is added to the KG, this set of paths might increase. It is easy to see that the set $\mathcal{P}_n(e_c, r_q)$ is updated *iff* a new path p_{new} of length $\leq n$ appears between e_c and e_t. In this case, the edges in p_{new} would form a cycle with the edge (e_c, r_q, e_t). The length of the cycle would be at most $len(p_{new}) + 1$ which in turn is at most of length $n+1$. This, to find entities for which the prior has changed after the addition of a new edge/entity, it is sufficient to find entities lying on cycles of length up to $n+1$ starting from the new entity/edge.

This mechanism for finding entities for recomputation is only approximate when computing the precision. We see from Eq 3, that the numerator depends on paths that lead to the answer entity (as with prior) while denominator depends on all n length paths around e_c. So, if the numerator is ever to be increased, we would catch that update by the proposed cycle finding method. However, even if an entity does not lie on a cycle with the new edge/entity, if there is a path of length n from e_c to the new edge/entity, the denominator count would

	People	Professions	Sports Org.	Religious Entities
At time $t-1$	Marvin Gay Shaquille O'Neal Avril Lavinge Woody Harrleson	Statistician Assoc. football manager Structural Engineer Financial backer	St. Louis Blues Orlando Pirates Sheffield Wednesday FC Malaya national football team	Isalm Russian Orthodox church Buddhism United Church of Christ
At time t	Elliot Smith Barbara Stanwick	Harpsichordist Child Actor	Excelsior Rotterdam Seattle Super Sonic	The Mormons Eastern Rite Catholic

Table 7: Example Clusters discovered in online setting. We show the assignment of new entities to the clusters in the particular time step (below line).

	WN18RR	FB122	NELL-995
HITS@1	0.422	0.694	0.296
HITS@3	0.461	0.739	0.405
HITS@10	0.508	0.779	0.502
MRR	0.451	0.724	0.367

Table 8: Results on Validation set

	WN18RR	NELL-995
HITS@1	41.8 $\pm$ (5.7e-2)	76.5 $\pm$ 2e-1
HITS@3	46.5 $\pm$ 0	85.2 $\pm$ 7e-2
HITS@10	51.3 $\pm$ (5.7e-2)	89.5 $\pm$ 1.4e-2
MRR	45 $\pm$ (5.7e-2)	81.45 $\pm$ 2e-1

Table 9: Mean and Variance across different hyper-params

be incremented. Thus, the precision estimates for some entities might be an over-estimate of the path precision (had it been recomputed after new edges are added to the KB).

A.5 Example Clusters

Table 7 shows some example of new entities arriving and getting assigned to their respective clusters by GRINCH.

A.6 Reproducibility Checklist

Computing Infrastructure: All our experiments were run on a Xeon E5-2680 v4 @ 2.40GHz CPU with 128 GB RAM. No GPUs were needed for the experiments.

The results on the validation set are reported in table 8 and avg. of 3 runs are reported in table 9. The NELL-995 does not come with a validation set, and therefore we selected 3000 edges randomly from the full NELL KB. As a result, many of the query relations were different from what was present in the splits of NELL-995 and hence is not a good representative. However, we report test results for the best hyper-parameter values that we got on this validation set.

The fixed number of parameters in our model are essentially the sparse non-learned entity vectors (which can be easily stored in COO format without taking much space). Other than that, our model is non-parametric with the number of parameters tied to the data.

For experiments on **WN18RR**:

- Inference time: 18.9 queries/s (total of 6268 queries)
- Train time: around 20 mins.
- Best Hyper-parameters:
  - Number of nearest-neighbor entities (K): 40
  - Number of paths from neighbors (N): 60
  - Max length of path (n): 5
  - Linkage for hierarchical clustering (λ): 0.25
- Hyper-parameter method / bounds: Grid search
  - K: [5, 10 , 15, 20, 30 , 40, 50]
  - N: [5, 10, 20, 40, 60 , 80]
  - λ: [0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6]

For experiments on **FB122**:

- Inference time:
- Train time: around 90 mins
- Best Hyper-parameters:
  - Number of nearest-neighbor entities (K): 10
  - Number of paths from neighbors (N): 80
  - Max length of path (n): 3
  - Linkage for hierarchical clustering (λ): 0.6
- Hyper-parameter method / bounds: Grid search

  - K: [5, 10 , 15, 20, 30 , 40, 50]
  - N: [5, 10, 15, 25, 60, 80]
  - λ: [0.4, 0.45, 0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.95]

For experiments on **NELL-995**:

- Inference time: 9.05 queries/s (total of 2825 queries)
- Train time: around 90 mins
- Best Hyper-parameters:
  - Number of nearest-neighbor entities (K): 15
  - Number of paths from neighbors (N): 25
  - Max length of path (n): 3
  - Linkage for hierarchical clustering (λ): 0.95
- Hyper-parameter method / bounds: Random search
  - K: [5, 10 , 15, 20, 30 , 40, 50]
  - N: [5, 10, 20, 40, 60, 80]
  - λ: [0.4, 0.45, 0.5, 0.6, 0.65, 0.7, 0.75]