ActiveLink: Deep Active Learning for Link Prediction in Knowledge Graphs

Natalia Ostapuk
University of Fribourg
Fribourg, Switzerland
natalia.ostapuk@unifr.ch

Jie Yang*
University of Fribourg
Fribourg, Switzerland
jie.yang@unifr.ch

Philippe Cudré-Mauroux
University of Fribourg
Fribourg, Switzerland
pcm@unifr.ch

ABSTRACT

Neural networks have recently been shown to be highly effective at predicting links for constructing knowledge graphs. Existing research has mainly focused on designing 1) deep neural network models that are expressive in capturing fine-grained semantics, e.g., NTN and ConvE, but that are however less scalable; or 2) shallow models that are scalable, e.g., TransE and DistMult, yet limited in capturing expressive semantic features. In this work, we demonstrate that we can get the best of both worlds while drastically reducing the amount of data needed to train a deep network by leveraging active learning.

We present a novel deep active learning framework, ActiveLink, which can be applied to actively train any neural link predictor. Inspired by recent advances in Bayesian deep learning, ActiveLink takes a Bayesian view on neural link predictors, thereby enabling uncertainty sampling for deep active learning. ActiveLink extends uncertainty sampling by exploiting the underlying structure of the knowledge graph, i.e., links between entities, to improve sampling effectiveness. To accelerate model training, ActiveLink further adopts an incremental training method that allows deep neural networks to be incrementally trained while optimizing their generalizability at each iteration. Extensive validation on real-world datasets shows that ActiveLink is able to match state-of-the-art approaches while requiring only 20% of the original training data.

CCS CONCEPTS

• **Theory of computation → Active learning**; • **Computing methodologies → Reasoning about belief and knowledge**; **Semantic networks**;

KEYWORDS

Deep active learning; neural link prediction; incremental training

ACM Reference Format:
Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. 2019. ActiveLink: Deep Active Learning for Link Prediction in Knowledge Graphs. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313620

*Corresponding author

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313620

1 INTRODUCTION

Knowledge graph construction is a key application for transforming Web content into machine-processable data [9]. A fundamental task in knowledge graph construction is link prediction, where the goal is to create or recover missing links in knowledge graphs, e.g., identifying the birthplace of a person or the CEO of a company. This task is generally formulated as a statistical relational learning problem, for which a variety of techniques such as latent factor models [25], random walks [19] and neural networks [22] have been explored. Among them, neural network-based methods, which learn semantic representations of entities and relations, have significantly advanced the state of the art in the past few years [5, 8, 30, 33, 38].

A focal point of research efforts on neural network-based methods has been designing models that are scalable enough to be applied to large knowledge graphs (e.g., millions of facts), while being able to precisely capture the semantics of entities and relations. Shallow models, such as translational distance models (e.g., TransE [5] and TransH [36]) and semantic matching models (e.g., DistMult [38], ComplEx [33]), were proposed to leverage simple algebraic operations – e.g., inner product and matrix-vector product – for modeling interactions among entities and relations in learning their representations [35]. While being scalable, these models are intrinsically lacking expressive power and thus are limited in capturing fine-grained semantics [8]. In contrast, deep models [9, 30] are better at learning expressive features. However, these models are more complex in terms of parameters. Consequently, model training requires very large amounts of data, which are rarely available in real-world scenarios [35].

Few studies have tackled this problem from the perspective of *data selection*. Better selecting training data is not only useful for reducing the size of the training data, thereby improving the efficiency of model training, but also beneficial for reducing the cost in acquiring additional data to improve link prediction performance. In this work, we propose to adopt active learning [7, 13, 27], which allows the model to choose the data from which it learns best. In the active learning setting, models initially trained on a small amount of data actively select the most informative data samples (often based on model uncertainty), which are then used in combination with existing training data to *retrain* the model. With such a selecting-retraining process iterating multiple times, the model can reach *state-of-the-art performance* with *significantly less training data* compared to conventional supervised learning settings.

The idea of combining deep learning models and active learning, i.e. *deep active learning*, has recently drawn attention for image [13, 16, 34] and text classification [29, 40]. Few approaches have however considered deep active learning for link prediction in knowledge graphs. Unlike image and textual data, the underlying data for link

prediction is relational: entities are connected by relations. This inherently provides an additional source of signals for data sampling in deep active learning: entities that are close to each other in the knowledge graph are more likely to share some similarity, thus to be more redundant for data sampling in the active learning process. By taking advantage of the underlying data structure, it is therefore possible to improve the effectiveness of data sampling.

While active learning is sample-efficient, it can still be computationally expensive since it requires to retrain models from scratch at every iteration of the active learning process. This problem becomes significant for deep active learning as it typically takes much more time to train a deep learning model than traditional machine learning models. Reducing the computational complexity is thus particularly important in our context, where the size of the training samples can easily blow up to tens of thousands of samples (an order of magnitude larger than what is used for image classification [13]). A straightforward workaround would be to incrementally train the model using the newly selected data or combining it with existing training data [29]. However, these methods will potentially bias the model towards the small amount of newly selected data, or towards the data selected in early iterations of the process. How to incrementally train deep link prediction models with optimal generalizability hence remains a key open research question.

In this paper, we introduce ActiveLink, a novel deep active learning framework for knowledge graphs that takes advantage of the underlying graph structure to improve the sampling effectiveness and to allow deep models to be incrementally trained in an unbiased manner. Our framework inherits from recent advances in Bayesian deep neural networks [12, 37] and takes a Bayesian view on existing neural link predictors. By doing so, it allows any neural link predictor to take into account uncertainty, thus enabling uncertain sampling in a deep active learning setting. In order to fully exploit the graph structure, ActiveLink incorporates uncertainty sampling into a clustering algorithm, which allows to identify redundant pieces of data in the knowledge graph for more effective data sampling. To accelerate model training, ActiveLink further adopts a principled method for unbiased incremental training based on meta-learning [2, 10]. Specifically, at each active learning iteration, we update the model parameters using the newly selected data samples with the meta-goal of generalizing the model for future predictions, which is approximated by generalizing the model based on the samples selected in preceding iterations. To do so, we strike a balance between the importance of newly and previously selected data in order to reach an unbiased estimate of the model parameters.

In summary, we make the following key contributions:

- We present a unified Bayesian view on neural link prediction, which allows any neural link predictor to be used in an uncertainty-based deep active learning setting;
- We propose a new data sampling algorithm that takes advantage of both model uncertainty and the underlying graph data structure to improve data sampling effectiveness;
- We introduce a new meta-learning method to incrementally train deep neural networks for link prediction through deep active learning with optimized generalizability;
- We demonstrate the effectiveness of our approach through an extensive evaluation on real-world datasets. Overall, ActiveLink is able to match the state-of-the-art approach while requiring *significantly less* (20% only) training data.

To the best of our knowledge, this is the first work considering deep active learning for link prediction in knowledge graphs. Our work is an initial yet important step towards improving link prediction performance through the efficient creation of training data. By strategically picking which data samples are used to train neural link predictors, ActiveLink provides an approach to effectively refine large training data and offers a route to efficiently create high-quality data (e.g., through crowdsourcing [39]). The later is highly important for scenarios where data creation is expensive, e.g., knowledge graph construction for specific domains.

2 RELATED WORK

In this section, we first discuss related work on neural network-based methods for link prediction, then review recent advances in deep active learning.

2.1 Neural Link Prediction

Existing neural link prediction methods fall into two broad categories, i.e., deep and shallow models, depending on whether or not the network contains at least one hidden layer. In the following, we review representative techniques for each category and discuss the limitations these techniques suffer from.

Shallow Models. Translational distance models are typical shallow models. A representative model is TransE [5], which learns low-dimensional representations (i.e., embeddings) for both entities and relations in a knowledge graph by minimizing the distance between linked entities. Extensions such as TransH [36] and TransR [21] project embeddings onto relation hyperplanes/spaces to learn relation-specific representations of entities. Another line of work learns representations for entities and relations by considering the match of their semantic representations as the learning target. A representative model of that line is RESCAL [24], which learns embeddings of entities while representing relations as a matrix to model the pairwise interactions between latent factors of two entities. DistMult [38] extends such a method by restricting relation matrices to diagonal ones; while being efficient, it can only be applied for symmetric relations. ComplEx [33] overcomes this issue by adopting complex-valued embeddings to model asymmetric relations. HolE [23] uses cross-correlation to decrease the number of parameters in relation representation and allows for modeling asymmetric relations. All these methods model the interactions between entities and relations through simple operations, e.g., matrix-vector products between relation matrices and entity embeddings.

Deep Models. Instead of using simple algebraic operations, a different line of work considers the use of neural networks with hidden layers to model the complex interaction patterns between entities and relations. An early piece of work [30] proposes Neural Tensor Network (NTN), which matches the embeddings of paired entities and relations by considering both linear and non-linear mappings, including those represented by relation-specific matrices and tensors and a hidden layer with non-linear transformation. Similar

models include semantic matching energy (SME) [4] and multilayer perceptrons (MLP) [9], which simplify NTN by discarding tensor parameters to improve model scalability. ConvE [8] is a recent work demonstrating that employing a 2-dimensional convolution and hidden-layers makes it possible to improve the model expressiveness while keeping a *relatively* smaller number of parameters.

Shallow models intrinsically lack expressive power, making them incapable of capturing fine-grained semantics of entities and relations [8]. Deep models have more expressive power but often involve much more parameters; consequently, model training is not scalable for large knowledge graphs in real-world scenarios. For instance, NTN has a time complexity two order of magnitude greater than TransE [35]. Our work takes an orthogonal perspective to improve the training efficiency of deep models by reducing the amount of training data while maintaining excellent performance through deep active learning.

2.2 Deep Active Learning

While extensive research has been carried out on both deep learning and active learning, researchers only recently started to investigate active learning approaches for deep neural networks. In the following, we briefly review related work that converges to the current notion of deep active learning.

Active Learning. In the traditional active learning setting, the model selects unlabeled data samples which supposedly can provide the strongest supervision; these samples are labeled and then used to retrain the model. The potential benefit of a data sample is generally measured by the model's uncertainty in making predictions for that sample, i.e., the so-called *uncertainty sampling* [7, 20]. It can be instantiated through different acquisition functions, such as maximum entropy [28], BALD [15], and variation ratios [11]. Besides uncertainty sampling, additional criteria can also be taken into account, e.g., how well a data sample will reduce the estimate of the expected error [26], which attempts to select data samples that directly optimize prediction performance. Such a criterion, however, is less practical as it is generally difficult to have an analytical expression for the expected prediction error.

Deep Active Learning. A key obstacle in applying active learning to deep neural networks is the fact that neural networks only make deterministic predictions, making it challenging to represent model uncertainty. A popular workaround has been to employ an active learning model (e.g., a Gaussian process) which is kept separate from the neural network classifier [17]. A more consistent solution is to leverage recent developments in Bayesian deep learning, which can generally be categorized into two classes. The first one is based on stochastic gradient descent (SGD). Welling et al. [1, 37] show that by adding the right amount of noise to standard SGD, the parameter converges to samples from the true posterior distribution. The second class of methods is based on dropout, which is a technique originally proposed to prevent over-fitting in training deep learning models [14, 31]. Gal et al. [12] showed that when considering dropout during prediction in a similar way as for model training, the predictions are equivalent to sampling from the approximate true posterior distribution of the parameters, thus turning a deterministic predictive function into a stochastic (uncertain) one.

Recent Applications. Deep active learning was first applied to image classification [13, 16]. These works showed that the Bayesian approach significantly outperforms deterministic predictions in deep active learning across multiple uncertainty-based acquisition functions. Recently, a few studies have investigated deep active learning approaches to textual data, including sentence classification [40] and named entity recognition [29]. These applications demonstrate that deep active learning can considerably reduce the amount of data used to train the model.

To the best of our knowledge, we are the first ones to leverage deep active learning for link prediction. Unlike image and textual data, data samples in link prediction are connected by a set of links, which provides an additional source of signals that we take advantage of thanks to a new data sampling algorithm. Moreover, we introduce a new principled method for incremental training, which is particularly useful to improve the computational efficiency of deep active learning on link prediction.

3 THE ACTIVELINK FRAMEWORK

This section introduces our proposed framework for deep active learning, which we refer to as ActiveLink. We first introduce a unified Bayesian view on neural link prediction, which allows any neural link predictor to represent prediction uncertainty, thereby enabling uncertainty sampling in ActiveLink. Next, we introduce two key features of ActiveLink: 1) structured uncertainty sampling, a data sampling method that leverages both model uncertainty and the underlying structure of the knowledge graph to better approximate sample informativeness ; 2) meta-incremental training, a meta-learning approach that allows the link prediction model to be incrementally trained in new iterations of the active learning process by optimizing the generalizability of the model for future predictions.

Problem Statement. Throughout this paper we use boldface lowercase letters to denote vectors and boldface uppercase letters to denote matrices. We use capital letters (e.g., $\mathcal{P}$) in calligraphic math font to denote sets. Following this, we denote a knowledge graph by $\mathcal{G} = \{(s, r, o)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ is the entity set, $\mathcal{R}$ is the relation set, and where a triple (s, r, o) represents a relationship $r \in \mathcal{R}$ between a subject $s \in \mathcal{E}$ and an object $o \in \mathcal{E}$.

Given a budget B, our goal is to select B triples from $\mathcal{G}$ to train a deep neural link predictor with optimal performance. We consider an active learning approach to this problem, which decomposes the problem into multiple iterations. At every iteration, we select k ($k < B$) triples from $\mathcal{G}$, and update the parameters of the link predictor using these triples in combination with previously selected triples. We consider data sampling performed in a greedy fashion, that is, at each iteration we select only k triples with the highest informativeness as determined by ActiveLink. The key problem for ActiveLink is therefore two-fold. 1) Effectiveness: selecting the most informative triples at each iteration, which is important for training a link predictor with high accuracy. 2) Efficiency: training the link predictor in an incremental fashion. Effectiveness and efficiency are addressed by structured uncertainty sampling and meta-incremental training, respectively, as we introduce next.

3.1 Uncertainty Sampling

We start by introducing a unified Bayesian view on neural link predictors. Based on this abstraction, we then introduce an uncertainty sampling method for deep active learning of neural link predictors.

Neural Link Predictors as Scoring Functions. Neural link predictors are often viewed as scoring functions $f_r(s, o)$, which in our context take as input a potential triple (s, r, o), and output a score representing the plausibility of the triple being true. Taking this view, the classic MLP model [9] is defined as follows:

$$f_r(s, o) = \mathbf{w}^\intercal g([\mathbf{s}; \mathbf{r}; \mathbf{o}]\mathbf{W}) \tag{1}$$

where $\mathbf{s}$ and $\mathbf{o}$ are entity embeddings, and $\mathbf{r}$ is a relation embedding; these embeddings are concatenated (denoted by $[;]$) to be used as the input for a fully connected layer parameterized by the linear transformation matrix $\mathbf{W}$ and a non-linear function g (e.g., tanh); $\mathbf{w}$ is a linear transformation vector that takes as input the output of the fully connected layer to obtain the final score.

The state-of-the-art model Convolutional 2D Knowledge Graph Embeddings (ConvE) [8] is similarly defined as follows:

$$f_r(s, o) = g(vec(g([\bar{\mathbf{s}}; \bar{\mathbf{r}}] * w))\mathbf{W})\mathbf{o} \tag{2}$$

where $\bar{\mathbf{s}}$ and $\bar{\mathbf{r}}$ are 2D reshaping of the entity and relation embeddings, which are concatenated and used as an input for a 2D convolutional layer with filters w; vec is a vectorization operation that transforms the feature map given by the convolutional layer to a vector; $\mathbf{W}$ is the weight matrix and g is a non-linear function (rectified linear units (ReLU) [18] are used in the original work).

Neural Link Predictors as Bayesian Models. To allow for uncertainty sampling in deep active learning, we adopt the Bayesian approach to deep neural networks recently developed by Gal and Ghahramani [12]. To this end, we first reformulate neural link predictors as parameterized likelihood functions, such that the output represents the probability of a triple being true. We then pose a prior distribution on the parameters, to transform a deterministic neural link predictor into a Bayesian model.

Denoting all parameters, including entity and relation embeddings and all weight parameters (i.e., linear transformation matrices and vectors), as Θ, we can rewrite any neural link predictor as $f_r^\Theta(s, o)$, which can then be reformulated as a likelihood function by introducing an additional softmax layer:

$$p(y|(s, r, o), \Theta) = \text{softmax}\,(f_r^\Theta(s, o)) \tag{3}$$

where y is the output of the link predictor representing whether a triple is true or not. To make neural link predictors Bayesian, we define a prior over the parameters Θ:

$$\Theta \sim p(\Theta|K) \tag{4}$$

e.g., a standard Gaussian prior parameterized by K (the co-variance matrix). With this definition, model training will result in a posterior distribution over the parameters, i.e. $p(\Theta|\mathcal{D}_{train})$ (where $\mathcal{D}_{train}$ is the training data), instead of point estimates (i.e., fixed values for the parameters). The prediction for an arbitrary input (s, r, o) can be described as a likelihood function:

$$p(y|(s, r, o), \mathcal{D}_{train}) = \int p(y|(s, r, o), \Theta)p(\Theta|\mathcal{D}_{train})d\Theta \tag{5}$$

which provides a more robust prediction than non-Bayesian methods as it takes into account the uncertainty of the parameters.

The problem of inferring the exact posterior distribution for the parameters, $p(\Theta|\mathcal{D}_{train})$, is intractable. Gal and Ghahramani [12] recently proposed in that context Monte Carlo (MC) dropout, which is a simple yet effective method for performing approximate variational inference. MC dropout is based on dropout [14, 31], which is typically used during model training to randomly drop hidden units of the network at each parameter update iteration; this reduces complex co-adaptations of neurons that can easily lead to overfitting. Gal and Ghahramani [12] prove that by performing dropout during the forward pass when making predictions, the output is equivalent to the prediction when the parameters are sampled from a variational distribution of the true posterior.

Formally, MC dropout is equivalent to sampling from a variational distribution $q(\Theta)$ that minimizes the Kullback-Leibler (KL) divergence to the true posterior $p(\Theta|\mathcal{D}_{train})$. Given this, we can perform a Monte Carlo integration to approximate Equation 5:

$$\begin{aligned} p(y|(s, r, o), \mathcal{D}_{train}) &\approx \int p(y|(s, r, o), \Theta)q(\Theta)d\Theta \\ &\approx \frac{1}{T}\sum_{t=1}^{T} p(y|(s, r, o), \widehat{\Theta}) \end{aligned} \tag{6}$$

where $\widehat{\Theta}$ is the parameters sampled T times from $q(\Theta)$, i.e., $\widehat{\Theta} \sim q(\Theta)$. To summarize, MC dropout provides a practical way to approximately sample from the true posterior without explicitly calculating the intractable true posterior.

Deep Uncertainty Sampling. Active learning aims at selecting the most informative data samples (triples, in our case) to train the model. This is often formulated as an acquisition function, defined as follows:

$$(s, r, o)^* = \underset{(s,r,o)\in\mathcal{G}}{\arg\max}\ \phi((s, r, o)) \tag{7}$$

The key to developing a good sampling strategy is designing an effective informativeness measure. With the Bayesian formulation of neural link predictors, the informativeness of a triple can be quantified by model uncertainty. A typical uncertainty measure is Shannon entropy:

$$\begin{aligned} &\phi((s, r, o)) = H[y|(s, r, o), \mathcal{D}_{train}] \\ &= -\sum_{C\in\{0,1\}} p(y = C|(s, r, o), \mathcal{D}_{train}) \log p(y = C|(s, r, o), \mathcal{D}_{train}) \\ &= -\sum_{C\in\{0,1\}} (\frac{1}{T}\sum_t \hat{p}_C^t) \log(\frac{1}{T}\sum_t \hat{p}_C^t) \end{aligned} \tag{8}$$

where $\frac{1}{T}\sum_t \hat{p}_C^t$ is the averaged predicted probability of class C for (s, r, o) ($C \in \{0, 1\}$ i.e., the triple being true or not), sampled T times by Monte Carlo dropout. Note that Θ is marginalized in the above equation as in Equation 6.

3.2 Structured Uncertainty Sampling

Besides model uncertainty, knowledge graphs provide an additional source of signals to represent data informativeness given their data structure: entities are linked by relations. Considering an existing pool of data sampled in preceding iterations, we make the following assumptions:

(1) Entities linked with those in the training pool are less informative than entities not linked;

(2) More generally, entities that are intensively linked with each other are less informative with respect to each other than those sparsely linked.

Relations therefore provide a source for measuring the informativeness of data samples from the perspective of the redundancy between data samples. This comes in contrast to representing informativeness from the perspective of the models. We are, therefore, interested in designing a data sampling method that exploits relations in knowledge graphs and that potentially combines it with model uncertainty.

To this end, we leverage clustering algorithms to identify groups of entities based on relations among entities. Our basic assumption is that entities belonging to the same cluster are more redundant with each other. Therefore, we construct our initial training set by picking from each of the c clusters a single data sample (i.e., a triple). After the model is trained on the initial set, at each of the following active learning iterations we select clusters based on their potential redundancy with respect to existing training data. Following the idea that data samples belonging to the same cluster are less informative, we select a sample from each of the selected clusters to form a new dataset for model training.

The above process is described by the high-level pseudo code in Algorithm 1. The key steps are the selection of clusters (row 10-12) and the selection of data samples from the clusters (row 13-15). For cluster selection, the redundancy score (row 11) is calculated as the averaged similarity between all data samples — specifically, cosine similarity between entity embeddings — in the cluster and those in the existing training pool. The selection of data samples within the cluster is based on model uncertainty (row 14). By doing so, we incorporate uncertainty sampling into the sampling algorithm.

We consider K-means for clustering entities, which are represented by low-dimensional embedding learned through TransE [5]. K-means allows picking the number of resulting clusters c. If c is less than the sample size k required for the active learning iteration, the sampling algorithm will be collapsed such that cluster selection is not considered and data samples will be selected from all clusters. In this case, we take the n_j most uncertain triples from cluster C_j, where n_j is proportional to the size of C_j.

We note that our algorithm is not restricted to any specific clustering algorithm. More advanced clustering methods can be applied, e.g., those based on relations among entities as well as entity attributes [3, 6]. Since our goal is to show that clustering can be an effective means to leverage the underlying structure of knowledge graphs for data sampling, we experiment on K-means as it is one the most widely used clustering methods. Comparison of different clustering methods is left for future work.

Algorithm 1: Structured Uncertainty Sampling

Input: Knowledge graph $\mathcal{G} = \{(s, r, o)\}$, budget B, #clusters c, #samples per iteration k ($k < B$), link predictor M
Output: Triples sampled at i-th iteration $\mathcal{T}_i$ ($i \in \{0, 1, \ldots\}$)

1 $C \leftarrow$ Cluster entities in $\mathcal{G}$;
2 $C^k \leftarrow$ Pick k clusters from C;
3 $\mathcal{T}_0 \leftarrow \emptyset$;
4 **foreach** $C_j \in C^k$ **do**
5 $(s, r, o) \leftarrow$ Pick a triple from C_j;
6 Add (s, r, o) to $\mathcal{T}_0$;
7 **while** $\sum_{l=1}^{i} |\mathcal{T}_l| \leq B$ **do**
8 Incrementally train M on $\mathcal{T}_i$;
9 $i++; \mathcal{T}_i \leftarrow \emptyset$;
10 **foreach** $C_j \in C$ **do**
11 Compute $redundancy\_score(C_j)$;
12 $C^k \leftarrow$ Pick k clusters based on the scores;
13 **foreach** $C_j \in C^k$ **do**
14 $(s, r, o) \leftarrow \arg\max\ \phi((s, r, o)), \forall s \in C_j$;
15 Add (s, r, o) to $\mathcal{T}_i$;

3.3 Meta-Incremental Training

In the traditional active learning setting, newly selected data samples are combined with existing training data to retrain the model from scratch. This is highly time-consuming when training deep models (our experiments show that model retraining from scratch takes 3 times longer than updating the model parameters from the previous iteration; see Section 4.3 for more details). We are, therefore, interested in incremental training of deep models using the newly selected data samples in every iteration of active learning.

A straightforward workaround is to fine-tune the model trained on the previous iterations using the new data samples only, which is a widely used approach in transfer learning. Such an approach, however, is likely to bias the model to the small amount of newly selected data. To prevent the model to get biased, recent work has proposed to combine the newly selected data with the existing training data and incrementally train the model with a small number of epochs [29]. This however, may not allow to fully exploit the newly selected data, compared with the data selected early that has already been used to update parameters in previous iterations. A central problem here is how to strike a good balance between the importance of new and previously selected data to reach an unbiased estimate of model parameters.

To solve this problem, we propose to adopt a meta-learning approach, where we update model parameters with the meta-goal that the updated model is most generalizable to future predictions. This is achieved in a two-step parameter updating scheme: we temporarily update model parameters with a standard gradient descent step using the newly selected data, followed by a meta-learning step where the learning algorithm performs a gradient descent step with the objective that model parameters will be more generalizable. The generalizability of the updated model parameters is approximated by the prediction loss on data samples selected in the current and the previous iterations.

Formally, in the i-th active learning iteration, model parameters learned in the previous iteration, denoted by Θ_{i-1}, are first updated to Θ'_i by a gradient descent step on the newly selected data $\mathcal{T}_i$:

$$\Theta'_i = \Theta_{i-1} - \alpha \Delta_\Theta \mathcal{L}(f_{\Theta_{i-1}}, \mathcal{T}_i) \tag{9}$$

where f denotes the model, α is the learning rate, $\mathcal{L}(f_{\Theta_{i-1}}, \mathcal{T}_k)$ is the loss function applied to the data samples of $\mathcal{T}_i$ given existing model Θ_{i-1}, and Δ_Θ is the gradient of Θ with respect to the loss.

Algorithm 2: Meta-Incremental Training

Input: The current iteration i, data selected in the current iteration $\mathcal{T}_i$, model parameter from the previous iteration Θ_{i-1}, window size w, data selected in iterations within the window $\{\mathcal{T}_{i-w}, \mathcal{T}_{i-w+1}, \ldots, \mathcal{T}_{i-1}\}$, learning rate α and β

1 **Output:** Updated model parameters Θ_i
2 **while** *True* **do**
3 **for** $l = i - w; l \leq i; l++$ **do**
4 Evaluate $\Delta_\Theta \mathcal{L}(f_{\Theta_l}, \mathcal{T}_l)$;
5 $\Theta'_l \leftarrow \Theta_{i-1} - \alpha \Delta_\Theta \mathcal{L}(f_{\Theta_{i-1}}, \mathcal{T}_l)$;
6 $\Theta_i \leftarrow \Theta_{i-1} - \beta \Delta_\Theta \sum_{l=i-w}^{i} \mathcal{L}(f_{\Theta'_l}, \mathcal{T}_l)$;
7 $\Theta_{i-1} \leftarrow \Theta_i$;
8 **if** *converged* **then**
9 break;

To avoid overfitting, we use a meta-learner that makes use of the triples selected in previous iterations within a certain time window w, i.e. $\{\mathcal{T}_{i-w}, \mathcal{T}_{i-w+1}, \ldots, \mathcal{T}_{i-1}\}$, and update the parameters such that the updated model generalizes well to data in these iterations:

$$\min_{\Theta} \sum_{l=i-w}^{i} \mathcal{L}(f_{\Theta'_l}, \mathcal{T}_l) = \sum_{l=i-w}^{i} \mathcal{L}(f_{\Theta_{i-1} - \alpha \Delta_\Theta \mathcal{L}(f_{\Theta_{i-1}}, \mathcal{T}_l)}, \mathcal{T}_l) \quad (10)$$

wherein Θ'_l represents the parameters updated with Equation 9 using data selected in the l-th iteration ($i - w \leq l \leq i$). With such an objective, the meta-learning step is then performed by stochastic gradient descent as follows:

$$\Theta_i = \Theta_{i-1} - \beta \Delta_\Theta \sum_{l=i-w}^{i} \mathcal{L}(f_{\Theta'_l}, \mathcal{T}_l) \quad (11)$$

where β is the meta-learning rate. Such a meta-learning step involves a second-order derivative (i.e., Hessian) with respect to the parameters, which is readily supported by some standard deep learning libraries.

The overall incremental training algorithm is described in Algorithm 2. We use mini-batch gradient descent for Meta-Incremental training, where mini-batches of data samples selected in the current and previous iterations are used in the inner loop of the meta-learning process (row 3-5) and different mini-batches of these iterations are used for meta-learning loops. Parameters are updated by the meta-learning step (row 6) several times until the current iteration converges.

4 EXPERIMENTS AND RESULTS

In this section, we report on a set of experiments we have conducted to evaluate the performance of ActiveLink.[1] We first evaluate our new data sampling method and the incremental training method of ActiveLink separately, by answering the following question:

- **Q1:** How effective are our uncertainty sampling and structured uncertainty sampling methods in determining the informativeness of data samples?
- **Q2:** How effective is our meta-incremental training method in speeding up the deep active learning process?

Subsequently, we evaluate ActiveLink as a whole by considering the following issues:

- **Q3:** How effective is ActiveLink compared with a traditional supervised learning method (i.e., non-active learning) for training neural link predictors?
- **Q4:** How do parameter settings of ActiveLink, including #samples per iteration and #clusters for data sampling and window size for incremental learning, affect the performance?

In addition, we investigate the generalizability of Activelink across neural link predictors and its scalability on a big dataset. In the following, we start by introducing our experimental setup, before answering each of the above questions in a separate subsection.

4.1 Experimental Settings

Dataset. We experiment on two publicly accessible datasets: FB15K-237 and WikiMovie:

- **FB15K-237** [32] is a subset of Freebase widely used for link prediction evaluation [5, 8, 36]. The knowledge graph mainly describes facts about sports, movies and actors. The original dataset FB15K was contributed by Bordes et al. [5]. Toutanova and Chen [32] noted that FB15K suffers from the test leakage problem: simple rule-based models can reach high prediction performance on the test triples by inverting triples in the training set. We use FB15k-237, a corrected version where inverse relations are removed.
- **WikiMovie** is a subset of Wikidata[2] that contains facts about movies such as directors, actors and genre. The original data is large but sparse. For example, 53% of the entities appear in only one triple, making it difficult to evaluate the link prediction models; by contrast, 95% of the entities in FB15K-237 have at least three training examples. As such, we filtered the dataset to keep a subset with only entities that appear in at least two triples. We prepare two versions of the WikiMovie dataset, namely WikiMovie-300K and WikiMovie-1M, which contain 300K and 1M triples, respectively.

We report key statistics of our datasets in Table 1. We note that WikiMovie-300K contains a comparable number of triples as FB15K-237 yet much more entities. Consequently, link prediction is more difficult on WikiMovie-300K. The comparison of ActiveLink performance on these two datasets, therefore, helps to investigate the impact of data sparsity on ActiveLink performance. WikiMovie-1M is used below to study the scalability of the ActiveLink framework in terms of reducing the amount of data needed to train neural link predictors on large knowledge graphs.

Comparison Methods. To demonstrate the effectiveness of both model uncertainty and data redundancy (considering the underlying data structure), we compare the following data sampling methods:

- **Random**, which randomly selects data samples at every active learning iteration;
- **Structured**, a variant of our method that randomly selects triples from each cluster (similar to stratified sampling);

[1] Our code and data are available at https://github.com/eXascaleInfolab/ActiveLink.

[2] https://www.wikidata.org/

Table 1: Descriptive statistics of the datasets.

	#Entities	#Relationships	#Triples
FB15K-237	14,541	474	310,116
WikiMovie-300K	36,001	588	286,683
WikiMovie-1M	104,500	788	987,896

- **Uncertainty**, a variant of our method that selects data samples based on Shannon entropy (Equation 8);
- **Structured-Uncertainty**, which selects data samples based on both model uncertainty and data redundancy (our proposed Algorithm 1).

We note that Uncertainty is a method that has been investigated in image recognition [13]; however, to the best of our knowledge, we are the first to apply it to link prediction in knowledge graphs.

To investigate the effectiveness of our proposed incremental training method, we compare the following model training techniques for deep active learning:

- **Retrain**, the conventional way of performing active learning, which trains the model from scratch at each new iteration;
- **Incremental** [29], the baseline incremental training method that combines newly and previously selected data to incrementally train the model from the last iteration;
- **Meta-Incremental**, our incremental training method that adopts meta-learning to get an unbiased estimate of the model parameters (Algorithm 2).

We evaluate the above model training methods in terms of both model performance and training efficiency.

Finally, to show the superiority of ActiveLink in model training over traditional supervised learning settings as well as to demonstrate the generalizability of ActiveLink on different neural link predictors, we apply both training settings to two state-of-the-art deep neural link predictors (see Section 3.1 for details):

- **ConvE** [8], a model that uses two-dimensional convolutions over entity and relation embeddings for link prediction;
- **MLP** [9], a multi-perceptron model using fully-connected layers over entity and relation embeddings.

Parameter Settings for Link Predictors & ActiveLink. The parameters of the link predictors and ActiveLink are empirically set based on a held-out validation set that contains 10% of the original data. For the link predictors (i.e., ConvE and MLP), we apply grid search in {20, 50, 100, 200} for the dimension of the embeddings and in {0.1. 0.3, 0.5} for the dropout on different layers of the prediction model. We select learning rates from {0.0001, 0.001, 0.01, 0.1, 1} and batch size from {64, 128, 256}. In particular, we use 3×3 filters for the convolution in ConvE. For ActiveLink, the number of samples per iteration is selected from {500, 1000, 5000, 10000}. The number of clusters for the Structured-Uncertainty sampling method is selected from {10, 100, 1000, 10000}. The window size for our Meta-Incremental method is selected from {1, 5, 10, 20, inf}, where inf indicates that all preceding iterations are considered for meta-learning.

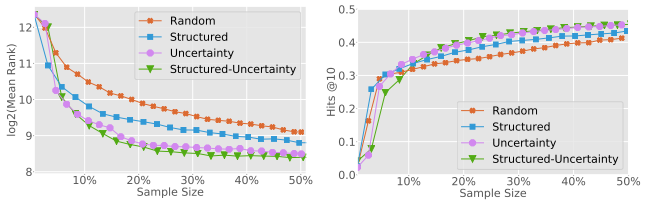
Evaluation Protocols. Evaluation is performed on a test set that contains 10% triples from the corresponding dataset; the remaining data is used as the training and validation sets. Following previous studies [5, 8, 21], we measure the performance using the following two metrics: 1) Mean Rank of correct entities, and 2) proportion of correct entities in the top-10 ranked entities (Hits@10). Lower Mean Rank and higher Hits@10 indicate better performance.

4.2 Comparative Results on Data Sampling (Q1)

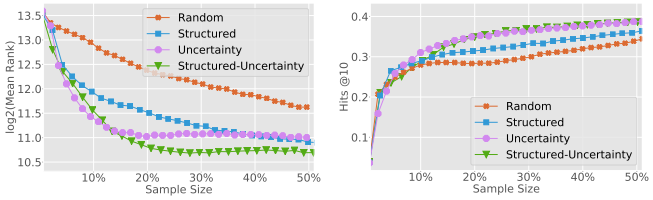
We start by investigating the effectiveness of our proposed sampling methods (Uncertainty, Structured, and Structured-Uncertainty) by comparing them against random sampling (Random). Results are reported in Figure 1, where performance is shown as a function of the fraction of training data (from zero to 50%) selected at different iterations of the active learning process.

From the figure, we observe that Random sampling is outperformed by Structured sampling and Uncertainty sampling across the two datasets and across the two performance metrics. This clearly demonstrates the effectiveness of model uncertainty and data redundancy in data sampling. Among Structured and Uncertainty sampling, we observe that Structured sampling generally performs better than Uncertainty sampling in the early iterations of the active learning process (before 5% samples are selected); however, in the late iterations, Structured sampling is outperformed by Uncertainty sampling. Such a pattern can be explained by the evolution of the size of the samples during the active learning process. In the early iterations of the active learning process, the size of the data being sampled is big as compared with the size of data samples already selected; it is, therefore, easy to select non-redundant data samples by considering the underlying data structure; when the size of selected data samples increases, the newly selected samples get increasingly more redundant with the selected samples, thus decreasing the effectiveness of Structured sampling. In comparison, model uncertainty is less affected by the size of the selected data samples. We note that, when the size of the selected data samples increases (30% to 40%), the performance gains of the link prediction model decreases even for Uncertainty sampling. This indicates the decreased utility of the data samples in the late iterations for improving model performance (see the next part of the results for additional evidence on that point).

Structured-Uncertainty sampling, which combines both model uncertainty and data redundancy in data sampling, achieves the best performance across the two datasets and the two performance metrics. The performance gains are most obvious for the WikiMovie-300K dataset measured by Mean Rank, as shown in Figure 1(c). We note that while not visually obvious in the other subfigures (due to the need for depicting the full range of the variations), Structured-Uncertainty consistently outperforms Uncertainty in every iteration as measured by both metrics: overall, the average improvements are 29.37 (Mean Rank) and 0.1% (Hits@10) for the FB15K-237 dataset when the sample size is between 40% and 50%; for the WikiMovie-300K dataset with the same sample sizes, the average improvements are 400.77 (Mean Rank) and 0.1% (Hits@10). These results highlight the difference between model uncertainty and data redundancy as data informativeness criterion and the effectiveness of combining both of them for data sampling.

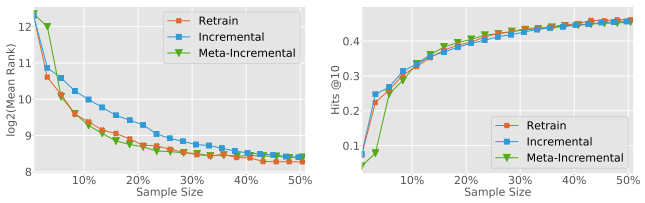


(a) FB15K-237 - Mean Rank (b) FB15K-237 - Hits@10

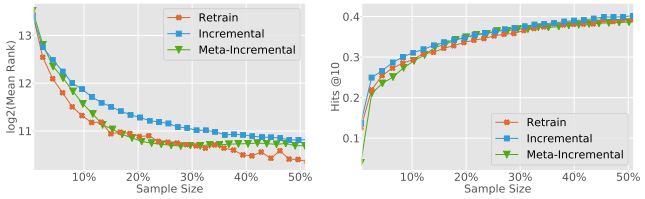


(c) WikiMovie-300K - Mean Rank (d) WikiMovie-300K - Hits@10

Figure 1: Comparison between data sampling methods. Upper figures compare the performance of the data sampling methods measured by (a) Mean Rank (semi-log scale) and (b) Hits@10 on the FB15K-237 dataset; lower figures compare the performance by (c) Mean Rank (semi-log scale) and (d) Hits@10 on the WikiMovie-300K dataset.



(a) FB15K-237 - Mean Rank (b) FB15K-237 - Hits@10



(c) WikiMovie-300K - Mean Rank (d) WikiMovie-300K - Hits@10

Figure 2: Comparison between model training methods. Upper figures compare the performance of the model training methods measured by (a) Mean Rank (semi-log scale) and (b) Hits@10 on the FB15K-237 dataset; lower figures compare the performance by (c) Mean Rank (semi-log scale) and (d) Hits@10 on the WikiMovie-300K dataset.

Compared with Random sampling, Structured-Uncertainty improves link prediction performance by 250.13 (Mean Rank) and 4.5% (Hits@10) for the FB15K-237 dataset and by 1710.04 (Mean Rank) and 5.2% (Hits@10) for the WikiMovie-300K dataset when 50% data samples are selected in both datasets.

4.3 Comparative Results on Incremental Training (Q2)

We now compare our Meta-Incremental training method with Retrain and the Incremental training baseline. Among them, Retrain should perform best in terms of model performance as Retrain uses all data selected in the new and previous iterations to reach an unbiased estimate of model parameters. Figure 2 shows the results.

From the results by Mean Rank in Figures 2(a,c), we observe that both Retrain and our Meta-Incremental training method outperform the Incremental baseline, and importantly, that our method performs as well as Retrain. This indicates that our proposed incremental training method indeed succeeds in obtaining an unbiased estimate of model parameters and achieves optimal model generalizability. The performance difference between our method and the Incremental baseline highlights the importance of including an unbiased estimation of the model parameters. We note that all three methods converge when the size of the data used in active learning reaches 50% of the size of the original training data, which also verifies the decreased utility of data selected in late iterations.

The above trends are not obvious from the results measured by Hits@10 in Figure 2(b,d): no clear difference is observed between the three model training methods. The different results measured by Mean Rank and Hits@10 implies that Hits@10 is a less sensitive metric in evaluating data sampling methods. This is likely due to the fact that Hits@10 is a set-based metric that does not take into account the ranking position of true positives in the *top-10* results, thus it is less sensitive in differentiating model performance, especially given the large number of entities in the experimental datasets.

Table 2: Model training time (in minutes).

	Retrain	Meta-Incremental
FB15K-237	499	177
WikiMovie-300K	945	334

Model Training Efficiency. In order to evaluate the runtime performance of our proposed Meta-Incremental training method, we investigate the model training time with up to 50% of the data samples selected using our test server[3], for both the Retrain method and our Meta-Incremental training method (both achieve optimal model performance). Table 2 reports the results. As shown by the table, our Meta-Incremental method requires a much shorter training time on both FB15K-237 and WikiMovie-300K: it speeds up model training by 2.8x in both cases.

In summary, our Meta-Incremental training method not only achieves optimal model generalizability, but also dramatically reduces the model training time.

4.4 ActiveLink vs. Non-active Learning (Q3)

To evaluate the effectiveness of ActiveLink as a whole, we compare the performance of neural link predictors trained with ActiveLink and those trained with a traditional supervised learning setting

[3] An Ubuntu 14.4 machine with a GeForce GTX TITAN X and 3.3 GHz CPU.

Table 3: Performance of ConvE and MLP trained by ActiveLink and the non-active learning setting on a varying fraction of the FB15K-237 and WikiMovie-300K datasets, measured by both Mean Rank and Hits@10.

Predictor	Fraction	FB15K-237				WikiMovie-300K			
		Mean Rank		Hits@10		Mean Rank		Hits@10	
		Non-Act.	ActiveLink	Non-Act.	ActiveLink	Non-Act.	ActiveLink	Non-Act.	ActiveLink
ConvE	10%	1325.26	409.14	0.255	0.403	6857.77	2150.08	0.219	0.329
	20%	822.88	351.98	0.282	0.440	5253.15	1641.98	0.259	0.365
	30%	605.79	339.48	0.301	0.450	4317.67	1673.97	0.271	0.379
	40%	537.12	329.21	0.326	0.459	3406.10	1658.99	0.296	0.388
	50%	458.91	318.18	0.349	0.464	2832.33	1628.06	0.314	0.396
MLP	10%	1386.80	487.64	0.248	0.395	7094.62	2520.71	0.216	0.300
	20%	848.21	374.75	0.283	0.440	5613.55	1536.02	0.248	0.357
	30%	663.18	346.94	0.314	0.457	4463.22	1379.29	0.274	0.376
	40%	547.16	326.11	0.332	0.467	3370.01	1326.25	0.298	0.389
	50%	458.51	322.40	0.354	0.470	2680.69	1372.14	0.325	0.398

using varying fractions of the experimental datasets. To demonstrate the generalizability of ActiveLink on different neural link predictors, we experimented with two neural link predictors, i.e., ConvE and MLP, and report the results in Table 3.

We observe that as the size of the training samples increases, the performance of ConvE and MLP also increases in both active learning and non-active learning settings. Importantly, ActiveLink consistently outperforms the traditional supervised learning setting across all experimental configurations in terms of the neural link predictor, the dataset, the fraction of the dataset for model training, and the performance metric. Such a result clearly indicates the superiority of ActiveLink in training neural link predictors when only a fraction of the dataset is used for model training. The performance gains achieved by ActiveLink flatten in the long run when most data samples are used for model training; this implies that the training set can no longer offer new informative data samples for improving model performance. Comparing the results on the two datasets, we observe that the performance gains through ActiveLink are larger for WikiMovie-300K than for FB15K-237, as measured by Mean Rank. Recalling that WikiMovie-300K contains 2.5x more entities than FB15K-237 does, these results suggest that ActiveLink brings more performance gains for sparse knowledge graphs.

We further compare the performance of ConvE and MLP trained by ActiveLink using a fraction of the dataset and by the supervised learning setting with the *full* dataset. Our experiments show that ActiveLink achieves nearly state-of-the-art results with only half of the data: for the FB15K-237 dataset, ActiveLink reaches 99.6% in Mean Rank and 96.4% in Hits@10 for ConvE, and 99.4% in Mean Rank and 94.6% in Hits@10 for MLP; for the WikiMovie-300K dataset, ActiveLink reaches 98.6% in Mean Rank and 97% in Hits@10 for ConvE, and 99.1% in Mean Rank and 97.4% in Hits@10 for MLP. These results strongly demonstrate the effectiveness of ActiveLink in data utilization when training neural link predictors.

Scalability for Large Datasets. In order to evaluate the performance of ActiveLink on large knowledge graphs, we compare it against a non-active learning setting on the WikiMovie-1M dataset. Results with ConvE as the neural link predictor are shown in Table 4 (similar results are observed with MLP). Overall, ActiveLink reaches 99% in MeanRank and 91% in Hits@10 with only **20%** of the data used for training. Compared with the previous results on the smaller datasets (FB15K-237 and WikiMovie-300K), these results show that ActiveLink is more effective in reducing the amount of training data required for large knowledge graphs.

Table 4: Performance of ConvE trained by ActiveLink using 20% of the WikiMovie-1M dataset and by the non-active learning setting using the full dataset.

	Mean Rank	Hits@10
ActiveLink	3356	0.337
Non-active learning	2341	0.371

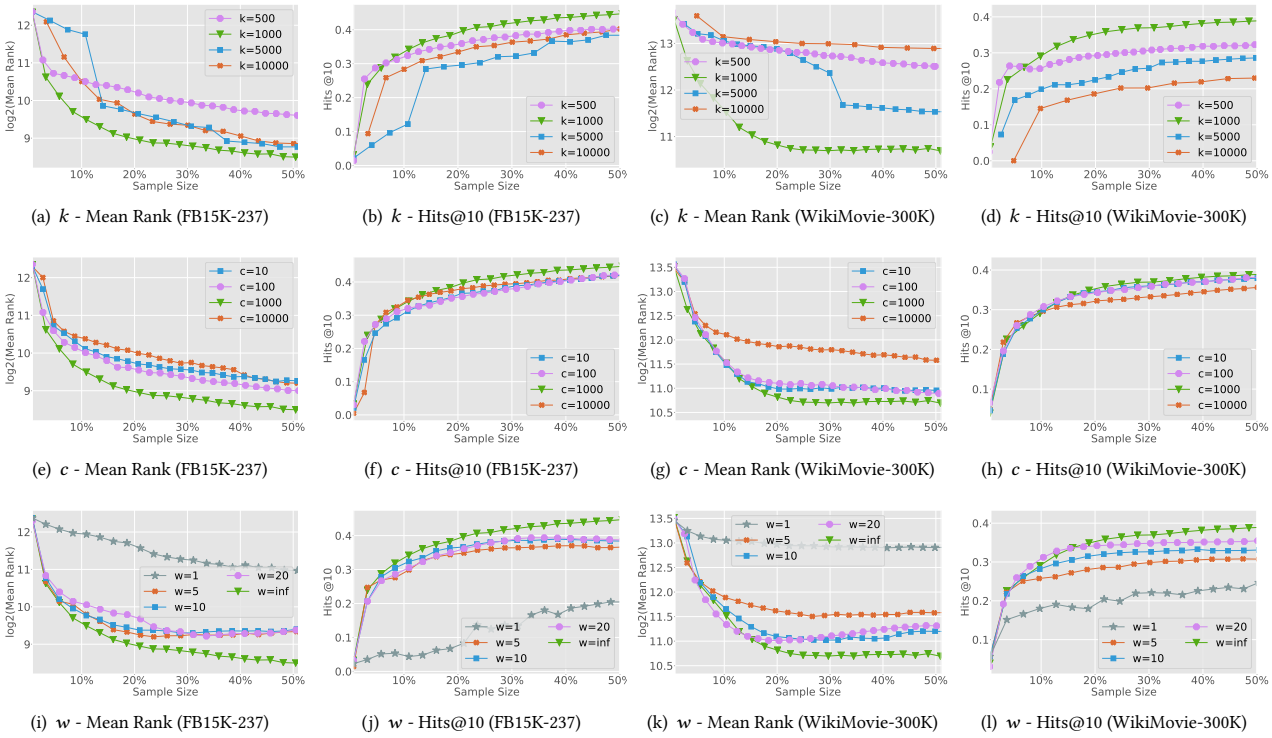
As a remark, we note that ActiveLink is less time efficient compared to the non-active learning setting due to the multiple training iterations. This, however, comes as a necessity to reduce the amount of data required for model training.

4.5 Parameter Sensitivity (Q4)

We now investigate the impact of parameters on the performance of ActiveLink. Figure 3 shows the performance of ActiveLink with different values for the number of samples per iteration (k), the number of clusters (c) and the window size w on the FB15K-237 and WikiMovie-300K datasets.

Impact of k. Due to the high requirement in terms of training data by deep models, the sample size k of individual iterations is supposed to be bigger in deep active learning settings than in traditional active learning settings. However, when the sample size gets too large, the benefits of increased data samples might be less significant than the benefits of training a better prediction model using fewer data samples and using that model to select new data samples for the next iteration. Figures 3(a-d) confirm this hypothesis: as the sample size increases, the performance first increases then decreases. The optimal performance is achieved when k is set to 1000.

Impact of c. The number of clusters c in our proposed Structured-Uncertainty sampling method controls the strength of the influence



(a) k - Mean Rank (FB15K-237) (b) k - Hits@10 (FB15K-237) (c) k - Mean Rank (WikiMovie-300K) (d) k - Hits@10 (WikiMovie-300K)

(e) c - Mean Rank (FB15K-237) (f) c - Hits@10 (FB15K-237) (g) c - Mean Rank (WikiMovie-300K) (h) c - Hits@10 (WikiMovie-300K)

(i) w - Mean Rank (FB15K-237) (j) w - Hits@10 (FB15K-237) (k) w - Mean Rank (WikiMovie-300K) (l) w - Hits@10 (WikiMovie-300K)

Figure 3: Impact of (a) #samples per iteration k, (b) #clusters c and (c) window size w on the performance of ActiveLink on both FB15K-237 and WikiMovie-300K datasets, measured by Mean Rank (shown in semi-log scale) and Hits@10.

of the knowledge graph data structure on sampling. Structured-Uncertainty sampling with a small c relies more on the model's uncertainty when selecting data samples, while a large c relies more on the data structure of the knowledge graph. Figures 3(e-h) show that as the value of c increases, the performance first increases then decreases. The best performance is achieved when c is set to 1000, with a significant margin over the other settings. This result suggests that with an appropriate setting for the number of clusters, our proposed data sampling method can ideally combine model uncertainty with the underlying structure of the knowledge graph.

Impact of w. In the Meta-Incremental training approach, the window size w controls the size of the selected data samples for optimizing the generalizability of the incrementally trained model. A larger window will include more data from previous iterations when updating the model parameters, thereby better approximating the model's generalizability for future predictions. This intuition is confirmed in Figures 3(i-l). We observe that the performance of ActiveLink increases along with the increase of the window size. The model is biased to the latest iterations when the window size is too small (shown by the result of $w = 1$). The best performance is achieved when pieces of data from all preceding iterations are used for incremental learning. However, training with bigger window sizes takes more time. Therefore, in real applications, the selected window size would need to strike a balance between model generalizability and training efficiency. Interestingly, $w = 1$ (i.e., only data in the most recent iteration is used together with the current iteration for meta-learning of model parameters) is outperformed by all the other configurations by a large margin. Such low generalizability indicates that the model is biased to the latest iterations when the window size is too small.

5 CONCLUSION

We presented ActiveLink, a deep active learning framework for neural link prediction with optimized data sampling and model training. By taking a Bayesian view on neural link predictors, ActiveLink enables to use uncertainty sampling for deep active learning of link predictors. ActiveLink extends uncertainty sampling by taking advantage of the underlying data structure of the knowledge graph to improve sampling effectiveness. In addition, it adopts a meta-learning approach to incrementally train neural link predictors with high generalizability. We extensively evaluated our framework on three real-world datasets and showed that it can drastically reduce the amount of data needed to train a neural link predictor while reaching state-of-the-art performance. As future work, we plan to apply ActiveLink to solicit high-quality data from a crowd of annotators to further improve neural link prediction.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 683253/GraphInt).

REFERENCES

[1] Sungjin Ahn, Anoop Korattikara, and Max Welling. 2012. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380* (2012).

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*. 3981–3989.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008.

[4] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94, 2 (2014), 233–259.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*. 2787–2795.

[6] Lijun Chang, Wei Li, Lu Qin, Wenjie Zhang, and Shiyu Yang. 2017. pSCAN: Fast and Exact Structural Graph Clustering. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 29, 2 (2017), 387–401.

[7] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research (JAIR)* (1996).

[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 1811–1818.

[9] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 601–610.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* (2017).

[11] Linton C Freeman. 1965. *Elementary applied statistics: for students in behavioral science*. John Wiley & Sons.

[12] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 1050–1059.

[13] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 1183–1192.

[14] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580* (2012).

[15] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[16] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision?. In *Advances in Neural Information Processing Systems (NIPS)*. 5580–5590.

[17] Andreas Krause, Ajit Singh, and Carlos Guestrin. 2008. Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research (JMLR)* 9, Feb (2008), 235–284.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 1097–1105.

[19] Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 529–539.

[20] David D Lewis and William A Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 3–12.

[21] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2181–2187.

[22] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[23] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic Embeddings of Knowledge Graphs.. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 1955–1961.

[24] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Vol. 11. 809–816.

[25] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*. ACM, 271–280.

[26] Nicholas Roy and Andrew McCallum. 2001. Toward Optimal Active Learning Through Monte Carlo Estimation of Error Reduction. *Proceedings of the 18th International Conference on Machine Learning (ICML)* (2001), 441–448.

[27] Burr Settles. 2010. Active Learning Literature Survey. *University of Wisconsin, Madison* 52, 55-66 (2010), 11.

[28] Claude Elwood Shannon. 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5, 1 (2001), 3–55.

[29] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. *arXiv preprint arXiv:1707.05928* (2017).

[30] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*. 926–934.

[31] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)* 15, 1 (2014), 1929–1958.

[32] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1499–1509.

[33] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2071–2080.

[34] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. 2016. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology* (2016).

[35] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 29, 12 (2017), 2724–2743.

[36] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 14. AAAI, 1112–1119.

[37] Max Welling and Yee W Teh. 2011. Bayesian Learning via Sochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 681–688.

[38] 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 *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

[39] Jie Yang, Thomas Drake, Andreas Damianou, and Yoelle Maarek. 2018. Leveraging crowdsourcing data for deep active learning - an application: Learning intents in Alexa. In *Proceedings of the 2018 edition of The Web Conference (WWW)*. ACM, 23–32.

[40] Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active Discriminative Text Representation Learning.. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 3386–3392.