StATIK: Structure and Text for Inductive Knowledge Graph Completion

Elan Markowitz[1,2,†,*], **Keshav Balasubramanian**[1,*], **Mehrnoosh Mirtaheri**[1,2], **Murali Annavaram**[1], **Aram Galstyan**[1,2], **and Greg Ver Steeg**[1,2]

[1]University of Southern California
[2]USC Information Sciences Institute
[*]Denotes equal contribution
[†]Corresponding author: esmarkow@usc.edu

Abstract

Knowledge graphs (KGs) often represent knowledge bases that are incomplete. Machine learning models can alleviate this by helping automate graph completion. Recently, there has been growing interest in completing knowledge bases that are dynamic, where previously unseen entities may be added to the KG with many missing links. In this paper, we present **StATIK**–**St**ructure **A**nd **T**ext for **I**nductive **K**nowledge Completion. StATIK uses Language Models to extract the semantic information from text descriptions, while using Message Passing Neural Networks to capture the structural information. StATIK achieves state of the art results on three challenging inductive baselines. We further analyze our hybrid model through detailed ablation studies.

1 Introduction

Knowledge graphs (KGs) are appropriate representations of knowledge bases across many domains. These domains include commonsense reasoning (Bauer, 2021; Yan et al., 2021; Zhang et al., 2020), question answering (Yasunaga et al., 2021; Feng et al., 2020; Lin et al., 2019; Christmann et al., 2019; Saxena et al., 2021; Hixon et al., 2015), recommendation systems (Guo et al., 2020; Wang and Cai, 2020; Huang et al., 2018; Wang et al., 2018), and many others (Hogan et al., 2021). These graphs are extremely large and often incomplete. As a result, there is significant interest in training machine learning models that can help complete these knowledge bases. In knowledge graphs, the nodes, called *entities*, often possess textual descriptions, while edges are typically labeled with one of many *relation* types, which may also possess textual descriptions. Effective KG completion models should learn to leverage this textual information in order to correctly complete the knowledge base. Additionally, such knowledge graphs are usually dynamic (Das et al., 2018; Liao et al., 2021) as a result of the underlying knowledge base being dynamic. Typically, this manifests as nodes and edges being added and removed from the knowledge graph while the set of relation types remains more static. Thus, another quality we desire of knowledge graph completion models, is that they be inductive and generalize to unseen entities.

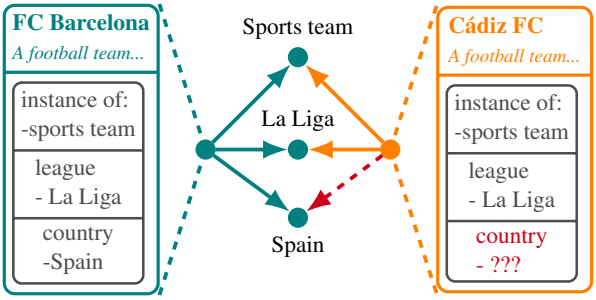


Figure 1: Depiction of the problem addressed by inductive learning. During training, only the **blue** portion of the graph exists, including the entities **FC Barcelona**, **sports team**, **La Liga**, and **Spain**. Later, the entity **Cádiz FC** is added to the graph. When added, an entity contains a description and some number of edges (possibly zero). Since StATIK is inductive, it requires no retraining or retroactive processing in any way to make predictions about Cádiz FC. This could include predicting Cádiz FC's country i.e. the query (Cádiz FC, country, ?). The correct prediction, (Cádiz FC, country, Spain), is displayed in dashed red.

We propose a completely inductive, hybrid model, called *StATIK*, that effectively leverages both the structure of a knowledge graph as well as the underlying textual descriptions of the entities and relations. Structure is incorporated through a Message Passing Neural Network (MPNN) (Gilmer et al., 2017) that aggregates information from a neighborhood defined around each entity, while textual information is incorporated through a pretrained language model such as BERT (Devlin et al., 2019). Our main contributions are summarized as follows:

1. We propose a completely inductive and hy-

Model	TransE	OpenWorld	Glove-DKRL	Commonsense	IndTransE	LAN	GraIL	KGBert	BLP	StAR	**ours**
Inductive - Seen2Unseen	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Inductive - Unseen2Unseen	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
End-to-end LM	✗	✗	✗	✗†	✗	✗	✗	✓	✓	✓	✓
No Support Set Required	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓
Graph features	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓
Structure Objective	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Inference Scalability	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(NQ)$	$\mathcal{O}(NQ)$	$\mathcal{O}(N)$	$\mathcal{O}(NR+Q)$	$\mathcal{O}(N+Q)$

Table 1: Related works comparison table. N is number of entities, Q is number of queries, R is number of relation types. †Model uses domain adaptation but does not train end-to-end. References are TransE (Bordes et al., 2013), OpenWorld(Shah et al., 2019), Glove-DKRL(Xie et al., 2016), Commonsense(Malaviya et al., 2020), IndTransE(Dai et al., 2021), LAN(Wang et al., 2019a), GraIL(Teru et al., 2020), KGBert(Yao et al., 2019), BLP(Daza et al., 2021), StAR(Wang et al., 2021)

brid knowledge graph completion model composed of an MPNN to leverage structure and a language model to leverage text.

2. We demonstrate empirically that incorporating structure via an MPNN leads to much better generalization in the inductive setting.

3. We achieve state-of-the-art results on three inductive benchmarks in which each predicted triple contains at least one new entity (dynamic setting) or exclusively new entities (transfer setting).

4. We design our model with scale in mind and show that the proposed model is significantly faster than similar alternatives, particularly for inference.

2 Inductive Representation Learning on Knowledge Graphs

We can define a knowledge graph with textual information as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{D})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relation types, $\mathcal{T}$ is the set of triples $(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, and $\mathcal{D}$ is the set of entity and relation descriptions. The inductive graph completion task is defined as follows. Let the training graph be $\mathcal{G}_{train} = (\mathcal{E}_{train}, \mathcal{R}, \mathcal{T}_{train}, \mathcal{D}_{train})$ where $\mathcal{E}_{train}$ is a subset of $\mathcal{E}$, $\mathcal{D}_{train}$ is the corresponding subset of $\mathcal{D}$, and $\mathcal{T}_{train}$ is the subset of $\mathcal{T}$ containing triples only involving entities in $\mathcal{E}_{train}$. The inference task is then to predict the triples in $\mathcal{T}/\mathcal{T}_{train}$, only having trained the model on $\mathcal{G}_{train}$. Specifically, for a given evaluation triple, $\mathcal{T}_i = (h, r, t)$, do head and tail prediction on the graph $\mathcal{G} - \mathcal{T}_i$. This means that given a *query* of the form $(h, r, ?)$ or $(?, r, t)$, rank all possible tail or head candidates (*targets*) such that the real triple is ranked as highly as possible. Figure 1 demonstrates a motivating example.

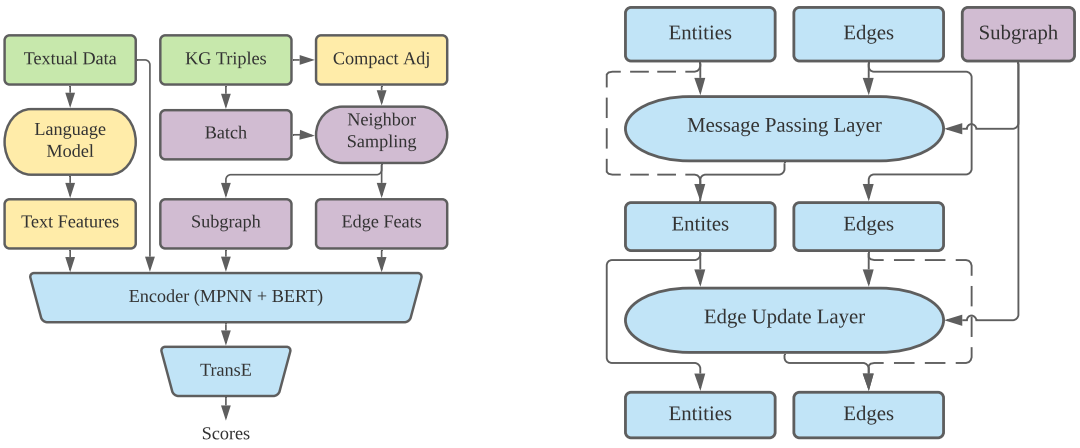
Similar to the transductive setting, such a task can be solved by learning a model that scores triples through minimizing some objective. We use a margin ranking loss as our objective where given a set of real triples, T, a corresponding set of negative triples, T', and a scoring function f (higher scores imply more likely triple), we compute the loss as

$$\sum_{(t,t') \in (T \times T')} max\left(0, 1 - f(t) + f(t')\right) \quad (1)$$

In order to learn the inductive objective, the model should avoid entity specific parameters such as an entity embedding tables as those parameters will not translate to the new entities at test time. To tackle this challenge `StATIK` uses text features instead of embeddings and extends prior work by also incorporating structural information through message passing.

3 Related Work

Much of the work in the area of Knowledge Graph Completion has focused on the transductive setting i.e. performing link prediction between entities seen at training time. Generally, these methods learn embeddings in a geometric space such as TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), DistMult (Yang et al., 2015), RotatE (Sun et al., 2019), and SimplE (Kazemi and Poole,



(a) Model Flowchart

(b) Message Passing Layer

Figure 2: Depiction of model flowchart (left) and the MPNN component (right). Green indicates model inputs. Yellow indicates pre-processing steps. Purple indicates data loading steps. Blue indicates model computation steps. Solid lines represent inputs and outputs. Dashed lines indicate residual connections.

2018), or through a machine learning decoder such as ConvE and HypER (Dettmers et al., 2017; Balazevic et al., 2018). There has also been effort in using graph neural networks for knowledge graph completion. R-GCN (Schlichtkrull et al., 2018) brings the original GCN (Kipf and Welling, 2017) to the multi-relational knowledge graph setting. Wang et al. (2020b) looked at using a modified version of GAT (Velickovic et al., 2018) to get strong results in the transductive setting.

3.1 Inductivity

Recently, there has been increased focus on the inductive setting. Works such as LAN (Wang et al., 2019a) and IndTransE (Dai et al., 2021) as well as a few others look at learning embeddings for new entities based on edges to entities in the training graph (Wang et al., 2020a; Bhowmik and de Melo, 2020). This requires a sufficient number of edges from nodes seen during training to the new nodes (**seen-to-unseen**). Other methods, such as GraiL (Teru et al., 2020), have been able to achieve inductivity without such requirements, and as a result, can operate on **unseen-to-unseen** entities.[1] There have also been some works on open domain knowledge graph completion, a similar learning task. These works include Shah et al. (2019) (OpenWorld) and Shi and Weninger (2018), and some of their techniques, such as using text to enable generalization to new entities, have continued in the works analyzed here. OpenWorld specifically aims to learn a function that aligns unsupervised text embeddings with knowledge graph embeddings so that new entities can be placed in the KG embedding space.

[1]GraiL is technically designed to do relation prediction, not link prediction, but is included here due to its relevance.

3.2 Support Sets

Most of the aforementioned inductive works (Wang et al., 2019a; Dai et al., 2021; Wang et al., 2020a; Bhowmik and de Melo, 2020) all require a support set, or edges connecting to known entities in the training graph, for the new entities seen at test time. While it is certainly useful to be able to use a support set when available, an ideal model would have the flexibility to use such edges when present but still be able to produce meaningful representations without them.

3.3 Language Models

Transformers (Vaswani et al., 2017), have created a renaissance in language modeling over recent years. Combined with self-supervised pretraining, language models such as BERT (Devlin et al., 2019) are able to capture the contextual and semantic information of natural language.

As many KGs contain text associated with each entity, researchers have sought to use that information for improved performance or inductivity. KGBert (Yao et al., 2019) looked into using transformers for link prediction, treating it as a text classification task. Bert for Link Prediction (BLP) (Daza et al., 2021) and StAR (Wang et al., 2021) have sought to incorporate language models while improving on some of the flaws of KGBert. Commonsense (Malaviya et al., 2020) also incorporates

a language model (along with a Graph Neural Network). However, it only uses the language model to initialize an embedding table. As a result, it is not inductive like the other models. Older model DKRL (Xie et al., 2016) uses a simpler language model with GloVe embeddings (Pennington et al., 2014).

3.4 Structural Objective

Most KG completion models use some form of structural objective; The scoring function uses spatial or geometric transformations to capture the graph structure. For instance, TransE applies the structural objective that a head embedding + a relation embedding should be close to the tail embedding of a true triple. KGBert is one of the few models that does not use such an objective.

3.5 Graph features

Structural objectives alone have some limitations with regard to capturing graph structure. Being able to explicitly use the local graph structure and topology as a feature (often through message passing) is beneficial for both general performance and inductivity. Many of the models mentioned (Malaviya et al., 2020; Dai et al., 2021; Wang et al., 2019a; Teru et al., 2020) make use of such features.

3.6 Scalability

Scalability is an incredibly important aspect of KG completion as knowledge graphs can include millions to billions of entities and edges. When dealing with complex encoders such as MPNNs or LMs, the number of encoder passes becomes an especially pressing issue, even for smaller graphs.

In KG completion, every entity is considered a possible solution to a query. If each of these possible triples is evaluated independently, the problem quickly becomes a combinatorial mess. This is the case for KGBert and GraiL which can only evaluate a single triple at a time. This makes the link prediction task quadratic in complexity. StAR improves on KGBert's approach but requires evaluating each entity in conjunction with every relation type. While definitely an improvement, this can still be problematic for big graphs.

Table 1 gives a comparison of the most relevant related works.

4 StATIK Architecture

We can conceptualize KG completion models as having an encoder and a decoder. Whereas most KG models use an embedding table for the encoder, our model utilizes a hybrid language model and MPNN based network. Our decoder is a relatively simple scoring function.

4.1 Language Models for Feature Extraction

For any entity or relation x, let $L(x) \in \mathbb{R}^{d_0}$ be the d_0 dimensional BERT-base encoding for the text associated with x. For every entity and relation type, we preprocess $L(x)$ for all x to create a meaningful feature matrix.

4.2 Encoder

We explore combining two different encoders, a transformer for processing text and a message passing neural network for processing structural data.

4.2.1 Language Model Transformer

Separately from the text featurization process, we also train through a language transformer in an end-to-end manner.

When encoding target entities, we simply pass the text associated with the entity through the encoder. However, when encoding queries, we condition the entity on the relation type. Similarly to StAR, for tail prediction–$(h, r, ?)$– we append the text associated with the head and the relation together giving `head_text + relation_text`. For queries of the form $(?, r, t)$, we prepend the relation text with the text "inverse of". This gives us the text for the query as `tail_text + "inverse of" + relation_text`. In contrast, StAR only ever appends head entities to relation text. This makes the head prediction task more complicated as one would have to encode each possible head-relation pair as the target of the head prediction queries.

4.2.2 Message Passing Graph Neural Network (MPNN)

We employ an MPNN of the following form (see fig 2b for overview). For each query of the form $(h, r, ?)$ or $(?, r, t)$, let s be the "query entity", meaning the head or tail that is part of the query (or the target entity if encoding target candidates). We then form a subgraph around s using the edges, E, connecting s to its neighbors, $\mathcal{N}(s)$.

We then compute the initial representations of all the entities in the subgraph, $V = \mathcal{N}(s) \cup \{s\}$, as

$$X_0 = \sigma\left(L(V)\mathbf{W_0^{(e)}}\right) \tag{2}$$

where $\mathbf{W_0^{(e)}}$ is a $\mathbb{R}^{d_0 \times d}$ parameter to reduce the dimension to the model's hidden dimension size d; σ is the model's element-wise activation function (LeakyReLU in our case).

For the m edges in the subgraph, let $\mathbf{i_h}$ and $\mathbf{i_t}$ be $\mathcal{E}^m$ vectors indicating the heads and tails respectively of each edge. Let $\mathbf{i_r} \in \mathcal{R}^m$ be the relation type of each edge. Let $\mathbf{i_{dir}} \in \{0, 1\}^m$ be the relative direction of the edge w.r.t. the query entity it is connected to, *s*. The initial edge representations are

$$E_0 = \sigma\left(L(\mathbf{i_r})\mathbf{W_0^{(r)}}\right) + \mathbf{E_{dir}}[\mathbf{i_{dir}}] \quad (3)$$

where $\mathbf{W_0^{(r)}}$ is a $\mathbb{R}^{d_0 \times d}$ parameter similar to $\mathbf{W_0^{(e)}}$, and $\mathbf{E_{dir}}$ is the binary embedding tables to encode the direction of the edge relative to the query entity.

4.2.3 Message Passing Layer

We follow the work of Galkin et al. (2020); Vashishth et al. (2020) and do not learn separate message passing transformations for each relation type. Instead, we learn only two transformations: One for messages passed in the forward direction and one for those passed backwards.

$$\begin{aligned} \mathrm{M}_{fwd} &= (X[\mathbf{i_h}] \parallel E)\mathbf{W_{fwd}} \\ \mathrm{M}_{back} &= (X[\mathbf{i_t}] \parallel E)\mathbf{W_{back}} \end{aligned} \quad (4)$$

where $\mathbf{W}_{<\mathbf{dir}>}$ is the $\mathbb{R}^{d \times d}$ weight matrix for the corresponding message direction, and $\parallel$ is the concatenation operator. The messages in M_{fwd} are sent to $\mathbf{i_t}$ while the messages in M_{back} are sent to $\mathbf{i_h}$. We then use mean aggregation and residual connections giving an entity update of

$$\begin{aligned} x_i' &= \frac{1}{|\mathcal{N}(i)|} \sum_{\mathcal{N}(i)} \mathrm{M}_{fwd}[\mathbf{i_t} = i] + \mathrm{M}_{back}[\mathbf{i_h} = i] \\ x_i' &= \sigma\left(x_i'\right) + x_i \end{aligned} \quad (5)$$

This calculation can be efficiently computed in parallel using vectorized `scatter_mean`.

4.2.4 Edge Update Layer

Similarly to (Gong and Cheng, 2019), we also maintain and update edge representations through each layer. We do this using a simple transformation using the previous edge representation and the entity representations of the head and tail:

$$E' = \sigma\left(\left(X[\mathbf{i_h}] \parallel E \parallel X[\mathbf{i_t}]\right)\mathbf{W^{(e)}}\right) + E \quad (6)$$

where $\mathbf{W^{(e)}}$ is the $\mathbb{R}^{3d \times d}$ edge update parameter. We use edge update layers between each message passing layer.

4.2.5 Combining the Encoders

We combine the the MPNN and the language model in a sequential manner, in which the output of the language model is used to replace the features for the entities being queried (query entities and target entities). These are the same entities whose representations will eventually be used by the decoder.

4.3 Decoders

We use a simple TransE (Bordes et al., 2013) model to score each candidate triple.

For given triple $q = (h, r, t)$, with final entity representations X, and relation embedding table $\mathbf{H} \in \mathbb{R}^{|\mathcal{R}| \times d}$, TransE scores the triple as

$$TransE(q) = -\|X[h] + \mathbf{H}[r] - X[t]\|_2 \quad (7)$$

where higher scores indicate a higher likelihood of existing in the graph. While, TransE, on it's own cannot represent all classes of relations (e.g. symmetric relations), this analysis does not apply when tied to a more expressive encoder.

4.4 Scalability

We aim to be able to use `StATIK` in real-world, large-scale settings. As a result, scalability is critically important. We employ a number of techniques that allow us to implement our message passing neural network on very large graphs.

4.4.1 Neighbor sampling

Neighbor sampling has been a key technique in scaling GNNs to large graphs (Hamilton et al., 2017b; Chen et al., 2018; bing Huang et al., 2018; Markowitz et al., 2021) and is even more critical with knowledge graphs. In homogeneous graphs, nodes have relatively similar degrees. In knowledge graphs, this is not the case. In our largest informally tested graph some nodes have degree greater than 10 million while the vast majority have degree less than 10. As a result neighbor sampling becomes a critical step to reduce computational complexity of the model.

		Train		Validation		Test	
	Relations	Entities	Triples	Entities	Triples	Entities	Triples
WN18RR	11	32,755	69,585	4,094	11,381	4,094	12,087
FB15k-237	237	11,633	215,082	1,454	42,164	1,454	52,870
Wikidata-5M	822	4,579,609	20,496,514	7,374	6,699	7,475	6,894

Table 2: Statistics of the datasets used in the experiments.

4.4.2 Compact Adjacency

In order to sample efficiently we adapt the compact adjacency structure from Markowitz et al. (2021) to the multi-relational setting. This data structure is similar to a CSR-formatted sparse matrix in which all the data is left-aligned. This allows fast row access, giving us the ability to get the neighbors of each node and sample from them in an efficient manner. Further details can be found in the appendix.

4.4.3 Query-Target Independence

As done in StAR and BLP, we separately calculate the embeddings for queries and targets, requiring $\mathcal{Q} + \mathcal{N}$ (#queries + #targets) passes through the encoder to run the graph completion task. However, unlike StAR, we do not need every entity-relation pair in order to do head prediction.

4.5 Optimized Negative Sampling

Rather than sampling negative entities as targets, that require additional passes through the encoder. We use the true targets for other queries in the batch as negative samples, filtering if the target would form a true triple. This does affect inference but does dramatically accelerate training.

5 Experiments

In this section we describe our thorough evaluation protocol. We describe the two settings in which we evaluate our inductive model and the infrastructure on which we implemented our model before demonstrating its effectiveness on three benchmark datasets. To analyze the importance of the various components of StATIK, we present the results of some detailed ablation studies. [2]

[2]The code for our work can be found at https://github.com/Elanmarkowitz/StATIK and more details for reproduction can be found in the supplement.

5.1 Evaluation Protocol

Our evaluation protocol follows that in (Bordes et al., 2013), wherein for each test triple (h', r, t') we generate two queries for the tail and head prediction tasks: $(h', r, ?)$ and $(?, r, t')$. For each query, we consider every entity in the graph as a candidate target. We then rank all candidate triples in decreasing order of score to see where the correct triple (h', r, t') ranks. We evaluate in the *filtered* setting as in Bordes et al. (2013). In the filtered setting, existing valid triples are ignored when ranking candidate targets. We report Mean Reciprocal Rank (MRR), Hits@1, Hits@3 and Hits@10 in line with literature. All our metrics are reported as the average between the head prediction and tail prediction tasks. This is more challenging than training for only one type of prediction at a time.

As in Daza et al. (2021), we demonstrate the effectiveness of StATIK in both a *dynamic* setting as well as a *transfer* setting. The dynamic setting corresponds to one in which at least one of the entities present in a given test triple (h', r, t') has not been seen by the model during training, i.e. $(h' \notin \mathcal{E}_{train} \vee t' \notin \mathcal{E}_{train})$. The dynamic setting represents the most likely way in which a knowledge graph grows over time. New entities that are added, connect to one or more existing entities as well as other new ones. The transfer setting represents the situation in which neither the head nor the tail already exists in the knowledge graph, i.e. $(h' \notin \mathcal{E}_{train} \wedge t' \notin \mathcal{E}_{train})$. This is a slightly more challenging setting and performance in this setting is a good indicator of inductivity.

5.2 Datasets

We test StATIK on three datasets. The first two are the inductive versions of WN18RR and FB15k-237 created in Daza et al. (2021). These datasets split the entities into a training set, $\mathcal{E}_{train}$, a validation set, $\mathcal{E}_{val}$, and a test set, $\mathcal{E}_{test}$. The training graph consists of the all the triples in which the head and tail are both in $\mathcal{E}_{train}$. The validation and test

Model	WN18RR				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Commonsense-KBC**	0.01	0.0055	0.009	0.019	0.00028	0.00001	0.000028	0.000056
GloVe-BOW*	0.170	0.055	0.215	0.405	0.172	0.099	0.188	0.316
BE-BOW*	0.180	0.045	0.244	0.450	0.173	0.103	0.184	0.316
GloVe-DKRL*	0.115	0.031	0.141	0.282	0.112	0.062	0.111	0.211
BE-DKRL*	0.139	0.048	0.169	0.320	0.144	0.084	0.151	0.263
BLP-TransE†	0.285	0.135	0.361	0.580	0.195	0.113	0.213	0.363
BLP-DistMult†	0.248	0.135	0.288	0.481	0.146	0.076	0.156	0.286
BLP-ComplEx†	0.261	0.156	0.297	0.472	0.148	0.081	0.154	0.283
BLP-SimplE†	0.239	0.144	0.265	0.435	0.144	0.077	0.152	0.274
StAR‡	0.321	0.192	0.381	0.576	0.163	0.092	0.176	0.309
StATIK	**0.516**	**0.425**	**0.558**	**0.690**	**0.224**	**0.143**	**0.248**	**0.381**
Improvement	**60.7%**	**121.3%**	**54.6%**	**19.8%**	**14.9%**	**26.5%**	**16.4%**	**5.0%**

Table 3: Inductive Link Prediction results on WN18RR and FB15k-237 in the *dynamic* setting. **Results of running (Malaviya et al., 2020) on our inductive data splits. *Baselines used in (Daza et al., 2021), † results of (Daza et al., 2021), ‡ Results of running (Wang et al., 2021) on our inductive data splits.

graphs given by the union $\mathcal{E}_{val} \cup \mathcal{E}_{train}$ or $\mathcal{E}_{test} \cup \mathcal{E}_{val} \cup \mathcal{E}_{train}$ respectively, and all triples contained within. The evaluation triples are those triples that were not present in the training data (nor in the validation set for the test triples). This is evaluation for the dynamic setting.

The third benchmark is Wikidata5M, curated and published by Wang et al. (2019b). Wikidata5M contains close to 5 million entities and 20 million triples. We use the inductive split intended by the authors of (Wang et al., 2019b), in which neither head nor tail in the validation or test triples have been seen during training. This is evaluation for the transfer setting. See Table 2 for an overview of the datasets.

5.3 Baselines

Since we are focused on the task of link prediction in the inductive setting, our main baselines are BERT for Link Prediction (BLP) (Daza et al., 2021) and StAR (Wang et al., 2021). Both recently proposed models that also seek to leverage textual information. For completeness, we also include the other baselines from Daza et al. (2021). These are a DKRL (Xie et al., 2016) implementation and a bag-of-words (BOW) model that represents an entity as the average of all its word embeddings. For these models, entity and relation features are created using GloVe embeddings as well as context free BERT embeddings. On Wikidata5M, we also compare against KEPLER (Wang et al., 2019b), the work that curated the dataset. To the best of our knowledge, these are the only inductive, embedding free, link prediction models, that also seek to jointly leverage structural and textual information.

Model	Wikidata-5M			
	MRR	H@1	H@3	H@10
KEPLER*	0.402	0.222	0.514	0.730
BLP-ComplEx†	0.489	0.262	0.664	**0.877**
BLP-SimpleE†	0.493	0.289	0.639	0.866
StATIK	**0.770**	**0.765**	**0.771**	0.779
Improvement	**56.1%**	**164.7%**	**16.1%**	-12.5%

Table 4: Inductive Link Prediction results on Wikidata-5M in the *transfer* setting. *Results of (Wang et al., 2019b), †Model variants that perform the best on Wikidata-5M from (Daza et al., 2021).

5.4 Results

We present three sets of results: (1) The main results, which comprise the performance of StATIK on the three benchmark datasets; (2) Ablation studies to understand how different choices in our model affect the overall performance; (3) Inference speed results to demonstrate utility. Results on FB15k-237 and WN18RR are averaged over 5 runs, while results on Wikidata-5M are averages over 2 runs.

5.4.1 Main Results

Tables 3 and 4 summarize the performance of StATIK in comparison to other state of the art baselines in the dynamic and transfer settings. StATIK sets a new state-of-the-art on MRR,

Model Variant		WN18RR			FB15k-237		
Use MPNN	Finetune LM	MRR	H@1	H@10	MRR	H@1	H@10
✗	✗	0.082	0.032	0.176	0.129	0.079	0.224
✗	✓	0.307	0.213	0.500	0.211	0.132	0.362
✓	✗	0.488	0.365	0.629	0.179	0.105	0.328

Table 5: Analyzing the importance of each component of the model to overall performance. As a reference, the model variant used to obtain results in table 3 uses the MPNN as well as finetunes the language model.

	Dataset	
Model	WN18RR	FB15k-237
KGBert	46000	87000
STAR	434	321
BLP	14	21
StATIK (ours)	**9.5**	**4**

Table 6: Inference time per query in milliseconds per Query (ms / q) comparison between inductive models. Lower is better.

Hits@1 and Hits@3 on all datasets, sometimes by a remarkably large margin and only underperforms on Hits@10 on Wikidata-5M. In general we notice diminishing benefits as we go from Hits@1 to Hits@10. As noted in Wang et al. (2021), Hits@1 is a weakness for textual encoding based paradigms. StAR and BLP include a structural objective to attempt to remedy this but are outperformed by our GNN based model. Hence it is not only important to incorporate structure but also do so effectively.

5.4.2 Ablation Studies

In addition to the main results we run two sets of ablation experiments. The first of these is to understand the influence each model component (MPNN or Language Model) has on the overall model performanc. These results are summarized in Table 5. The second set of ablation experiments demonstrate the effect maximum number of words allowed in the entity description has on StATIK.

6 Discussion

6.1 Inductivity

StATIK shows much greater ability to generalize to entirely new entities that are added to the knowledge graph as evidenced by the much superior performance compared to baselines i n both dynamic (Table 3) and transfer settings (Table 4). We hypothesize that this is due to the power of MPNNs to generalize to new nodes added to a graph as well as entirely new graphs (Hamilton et al., 2017a; Velickovic et al., 2018; Wang et al., 2020b). We also note the effectiveness of our model at Hits@1 in the transfer setting, with less drop from Hits@3 and Hits@10 than in the dynamic setting. This phenomenon remains to be more deeply explored but possible factors include the quantity of data, repeated structural patterns in the test set, or other peculiarities of this dataset, such as how the transfer graph was constructed.

6.2 Influence of Model Components

As shown in table 5, we find that while both the language model and MPNN are important to model performance, which matters more depends on the dataset being evaluated on due to differing graph structure. On Wordnet, the MPNN influences overall model performance more, possibly because the graph has more repeated structure. On Freebase, with richer text descriptions, the language model influences the overall model performance more.

6.3 Effects of Description Length

Unsurprisingly, more input data means better performance. However, there are diminishing returns to description length. This is consistent with both Wordnet and Freebase as demonstrated in Figure 3. Since increasing the max word length can increase the computational demand as a result of the lamguage model, there is a need to trade off between model and computational performance.

6.4 Inference Scalability

Inference scalability is paramount in any potentially useful knowledge graph completion model, especially when the link prediction task involves a large number of target entities. Our comparison with other state of the art, inductive knowledge graph completion models in Table 6 demonstrates that StATIK is much quicker at inference time

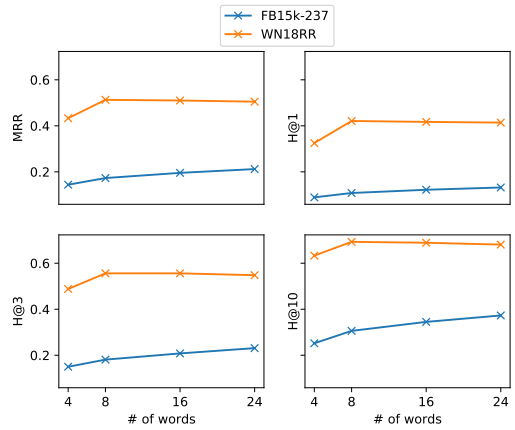


Figure 3: Max word length effect on performance.

than other competing models. These results are consistent with the theoretical computational complexity of each model (Table 1).

7 Conclusion and Future Work

Our work introduces and explores a new model for inductive knowledge graph completion that utilizes structural and textual information, and sets state-of-the-art on multiple benchmarks.

Ideas for future work include (i) testing this model with other types of graph neural networks that have already proved effective on KGs; (ii) testing alternative neighborhood sampling techniques beyond uniform random sampling; and (iii) extending this work to domains beyond text-based KGs.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0053, NSF MLWINS-2002874, USC-Meta Research Center, USC-ISI MINDS Group, and a gift from the Intel PrivateAI institute. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We would also like to acknowledge the reviewers who helped improve this work.

References

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2018. Hypernetwork knowledge graph embeddings. *CoRR*, abs/1808.07018.

Lisa Bauer. 2021. Identify, align, and integrate: Matching knowledge graphs to commonsense reasoning tasks. In *EACL*.

Rajarshi Bhowmik and Gerard de Melo. 2020. Explainable link prediction for emerging entities in knowledge graphs. In *SEMWEB*.

Wen bing Huang, Tong Zhang, Yu Rong, and J. Huang. 2018. Adaptive sampling towards fast graph representation learning. *ArXiv*, abs/1809.05343.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, J. Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Jian Jhen Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. *ArXiv*, abs/1801.10247.

Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.

Damai Dai, Hua Zheng, Fuli Luo, Pengcheng Yang, Baobao Chang, and Zhifang Sui. 2021. Inductively representing out-of-knowledge-graph entities by optimal estimation under translational assumptions. *ArXiv*, abs/2009.12765.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018. Building dynamic knowledge graphs from text using machine reading comprehension. *CoRR*, abs/1810.05682.

Daniel Daza, Michael Cochez, and Paul T. Groth. 2021. Inductive entity representations from text via link prediction. *Proceedings of the Web Conference 2021*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *EMNLP*.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. In *EMNLP*.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212.

Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9203–9211.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *ArXiv*, abs/2003.00911.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017a. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216.

William L. Hamilton, Zhitao Ying, and J. Leskovec. 2017b. Inductive representation learning on large graphs. In *NIPS*.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *NAACL*.

A. Hogan, E. Blomqvist, Michael Cochez, C. d'Amato, Gerard de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A. N. Ngomo, S. M. Rashid, A. Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and A. Zimmermann. 2021. Knowledge graphs. *Communications of the ACM*, 64:96 – 104.

Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

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

Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907.

Siyuan Liao, Shangsong Liang, Zaiqiao Meng, and Qiang Zhang. 2021. Learning dynamic embeddings for temporal knowledge graphs. WSDM '21, page 535–543, New York, NY, USA. Association for Computing Machinery.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *ArXiv*, abs/1909.02151.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Elan Markowitz, Keshav Balasubramanian, Mehrnoosh Mirtaheri, Sami Abu-El-Haija, Bryan Perozzi, G. V. Steeg, and A. Galstyan. 2021. Graph traversal with tensor functionals: A meta-algorithm for scalable learning. *ArXiv*, abs/2102.04350.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. 2021. Question answering over temporal knowledge graphs. In *ACL*.

M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and M. Welling. 2018. Modeling relational data with graph convolutional networks. *ArXiv*, abs/1703.06103.

Haseeb Shah, Johannes Villmow, Adrian Ulges, Ulrich Schwanecke, and Faisal Shafait. 2019. An open-world extension to knowledge graph completion models. *ArXiv*, abs/1906.08382.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *AAAI*.

Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *ArXiv*, abs/1902.10197.

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

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

Shikhar Vashishth, Soumya Sanyal, V. Nitin, and P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. *ArXiv*, abs/1911.03082.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, A. Casanova, Adriana Romero, P. Lio', and Yoshua Bengio. 2018. Graph attention networks. *ArXiv*, abs/1710.10903.

Baocheng Wang and Wentao Cai. 2020. Knowledge-enhanced graph neural networks for sequential recommendation. *Inf.*, 11:388.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748.

Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2020a. Entity context and relational paths for knowledge graph completion. *CoRR*, abs/2002.06757.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. *Proceedings of the 2018 World Wide Web Conference*.

Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2019a. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *AAAI*.

Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. 2020b. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access*, 8:5212–5224.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *CoRR*, abs/1911.06136.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and M. Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Jun Yan, Mrigank Raman, Tianyu Zhang, Ryan A. Rossi, Handong Zhao, Sungchul Kim, Nedim Lipka, and Xiang Ren. 2021. Learning contextualized knowledge structures for commonsense reasoning. *ArXiv*, abs/2010.12873.

B. Yang, Wen tau Yih, X. He, Jianfeng Gao, and L. Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *ArXiv*, abs/2104.06378.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *ACL*.

A Appendix

B Ethical Impact Statement

This work is designed for knowledge graph completion and thus may amplify the ethical ramifications of any work that utilizes such knowledge graphs. In addition, this work is designed to help automate the KG completion process. As a result, this may reduce the number of required employed maintainers of such knowledge bases. However, it also provides tools to increase the effectiveness of volunteer knowledge base maintainers. While impossible to predict the long term effects, we see many more clear positive near term effects than negative ones.

C Data Format

The datasets are structured as tab seperated files.

The triple splits (ind-train.tsv, ind-valid.tsv, ind-test.tsv) are formatted as
head_entity<tab>relation<tab>tail_entity

The entity description files (entity2text.txt, entity2textlong.tsv) are formatted as
head_entity<tab>description

The relation description files (relation2text.txt) are formatted as
relation<tab>description

D Reproducibility

D.1 Infrastructure

Experiments were conducted on a single GPU server equipped with 8 Nvidia RTX 5000 GPUs and an AMD EPYC 7502 32-Core Processor. All models are written using PyTorch (Paszke et al., 2019), and are trained using the GPUs in a data parallel fashion.

The GPU budget per run is 16 GPU-hrs for WN18RR, 24 GPU-hrs for FB15k-237, 480 GPU-hrs for Wikidata5M.

D.2 Hyperparameters

Minimal hyperparameter optimization was performed, instead, hyperparameters were copied

from those used in BLP (Daza et al., 2021). However, to take full advantage of the GPU's available, we increased the batch size to 32 training triples (64 training queries) per GPU and increased the learning rate from 2e-5 to 2e-4 to roughly compensate. We keep the epochs the same as in BLP (40 epochs for WN18RR/FB15k-237 and 5 epochs for Wikidata5M). Optimizer was AdamW with betas $(.9, .98)$ and linear schedule with warmup of 20% of training steps.

Sampled neighbors per entity was set to 10 for WN18RR and Wikidata5M, and tuned to 20 for FB15k-237 due to it's higher average degree.

We use the `bert-base-cased` version of BERT from HuggingFace (Wolf et al., 2020) as the language model for both feature extraction and training. This model has 110 million parameters, bringing the total number of parameters to 113 million.

LayerNorm is also used for the initial entity and relation representations as well as for each edge and entity updates in the MPNN.

We use 24 words of text for each input and hard-cap the number of tokens at 64. As per standard practice, special token `[CLS]` is used at the beginning of text, and `[CLS]` is used to separate entity and relation text as well as at the end of the tokenization.

E Limitations

The main limitation of the model is that it requires textual data for entities in order to run. It also does not work on graphs that constantly change the set of relation types.

Another limitation is that because it uses a transformer language model, it requires GPUs with good memory. The language model component contributes that vast majority of the weight of the model.