Commonsense Knowledge Base Completion with Structural and Semantic Context

Chaitanya Malaviya,[♢] Chandra Bhagavatula,[♢] Antoine Bosselut,[♢♣] Yejin Choi[♢♣]
[♢]Allen Institute for Artificial Intelligence
[♣]University of Washington
{chaitanyam, chandrab}@allenai.org, {antoineb, yejin}@cs.washington.edu

Abstract

Automatic KB completion for *commonsense* knowledge graphs (e.g., ATOMIC and ConceptNet) poses unique challenges compared to the much studied conventional knowledge bases (e.g., Freebase). Commonsense knowledge graphs use free-form text to represent nodes, resulting in orders of magnitude more nodes compared to conventional KBs ($\sim$18x more nodes in ATOMIC compared to Freebase (FB15K-237)). Importantly, this implies significantly sparser graph structures — a major challenge for existing KB completion methods that assume densely connected graphs over a relatively smaller set of nodes.

In this paper, we present novel KB completion models that can address these challenges by exploiting the structural and semantic context of nodes. Specifically, we investigate two key ideas: (1) learning from local *graph structure*, using graph convolutional networks and automatic graph densification and (2) *transfer learning* from pre-trained language models to knowledge graphs for enhanced contextual representation of knowledge. We describe our method to incorporate information from both these sources in a joint model and provide the first empirical results for KB completion on ATOMIC and evaluation with ranking metrics on ConceptNet. Our results demonstrate the effectiveness of language model representations in boosting link prediction performance and the advantages of learning from local graph structure (+1.5 points in MRR for ConceptNet) when training on subgraphs for computational efficiency. Further analysis on model predictions shines light on the types of commonsense knowledge that language models capture well.[1]

1 Introduction and Motivation

While there has been a substantial amount of work on KB completion for conventional knowledge bases such as Freebase, relatively little work exists for KB completion for *commonsense* knowledge graphs such as ATOMIC (Sap et al. 2019) and ConceptNet (Speer and Havasi 2013). The distinct goals of this paper are to identify unique challenges in commonsense KB completion, investigate effective methods to address these challenges, and provide comprehensive empirical insights and analysis.

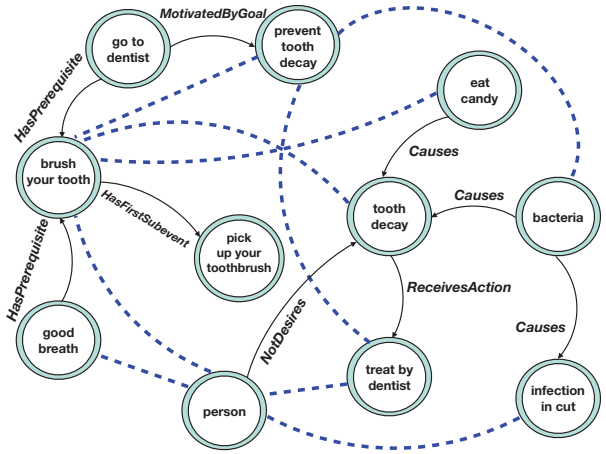

Figure 1: Subgraph from ConceptNet illustrating semantic diversity of nodes. Dashed blue lines represent potential edges to be added to the graph.

The key challenge in completing commonsense KGs is the *scale* and *sparsity* of the graphs. Unlike conventional KBs, commonsense KGs consist of nodes that are represented by *non-canonicalized*, *free-form* text, as shown in Figure 1. For example, the nodes "*prevent tooth decay*" and "*tooth decay*" are conceptually related, but *not* equivalent, thus represented as distinct nodes. This conceptual diversity and expressiveness of the graph, imperative for representing commonsense, implies that the number of nodes is orders of magnitude larger, and graphs are substantially sparser than conventional KBs. For instance, an encyclopedic KB like FB15K-237 (Toutanova and Chen 2015) has 100x the density of ConceptNet and ATOMIC (node in-degrees visualized in Figure 2).

In this work, we provide empirical insights on how the sparsity of commonsense KGs poses a challenge to existing KB completion models that implicitly assume densely connected graphs. Figure 3 provides a brief preview of this evidence, where the performance of ConvTransE (Shang et al.

[1]Code and dataset are available at github.com/allenai/commonsense-kg-completion.

2019), a high performing KB completion model, degrades quickly as we reduce the graph density of FB15K-237.

This motivates a strong need for investigating novel approaches to KB completion for commonsense KGs. We posit that new methods need to better accommodate the implicit conceptual connectivity across all nodes — both *structural* and *semantic* – beyond what is explicitly encoded in existing commonsense KBs. Specifically, we investigate two key ideas: (1) learning from local *graph structure*, using graph convolutional networks and automatic graph densification. and (2) *transfer learning* from language models to knowledge graphs to improve *contextual* representation of nodes.

To integrate graph structure information, we present an approach based on graph convolutional networks (GCN) (Kipf and Welling 2017) to contextualize a node's representation based on its local neighborhood. For transfer learning, we present effective approaches to fine-tune pre-trained language models (Devlin et al. 2019) to commonsense KGs, essentially achieving transfer learning from *language* to *knowledge*. Our work shares the high-level spirit of recent work from Petroni et al. (2019) that demonstrates the use of pre-trained LMs for reconstructing KB entries, but we provide a more focused study specifically for commonsense KGs. Empirical analysis leads to the observation that GCNs, although effective on various densely connected graphs (Schlichtkrull et al. 2018), are not as effective on commonsense KGs out of the box, as sparse connections do not allow effective knowledge propagation. Hence, we propose an approach for automatic graph densification based on semantic similarity scores between nodes. Finally, we highlight strategies necessary to train models using information from both the graph structure and language models.

Our main contributions are highlighted below:

1. Empirical insights about the unique challenges of commonsense KB completion compared to conventional encyclopedic KB completion.
2. Novel KB completion methods to model the implicit structural and semantic context of knowledge beyond what is explicitly available in existing KGs.
3. The *first* empirical results on ATOMIC for KB completion and evaluation with ranking metrics on ConceptNet.
4. Analysis and insights on types of commonsense knowledge captured well by language models.

In sum, our findings indicate that transfer learning is generally more effective than learning from graph structure. Moreoever, we find that graph structure can indeed provide complementary information which can boost performance, especially when training with subgraphs for efficiency.

2 Knowledge Graphs

There have been several efforts in building graph-structured representations of commonsense (Lenat 1995; Speer and Havasi 2013; Cambria, Olsher, and Rajagopal 2014; Sap et al. 2019). We focus our experiments on two prominent knowledge graphs: ConceptNet and ATOMIC. Statistics for both graphs are provided in Table 1, along with FB15K-237 – a standard KB completion dataset.

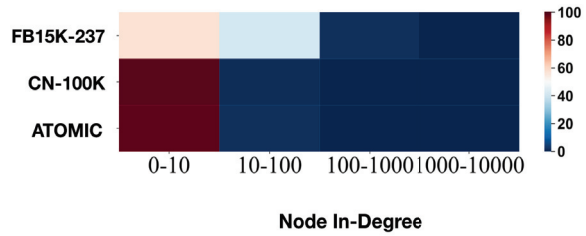

Figure 2: Heatmap showing the percentage of nodes with in-degree belonging to the specified bins on the x-axis. The plot illustrates the sparseness of commonsense KGs relative to a standard KB completion benchmark (FB15K-237).

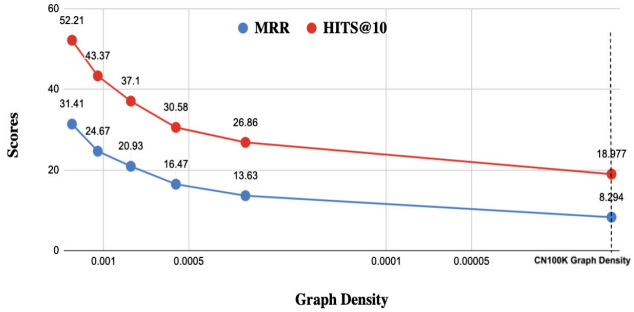

Figure 3: Trend of decreasing KB Completion Scores with different values of graph density (in log scale) for the FB15K-237 dataset with the ConvTransE model.

ConceptNet-100K[2]**:** CN-100K contains general commonsense facts about the world. This version (Li et al. 2016) contains the Open Mind Common Sense (OMCS) entries from ConceptNet (Speer and Havasi 2013). The nodes in this graph contain 2.85 words on average. We used the original splits from the dataset, and combined the two provided development sets to create a larger development set. The development and test sets consisted of 1200 tuples each.

ATOMIC[3]**:** The ATOMIC knowledge graph contains social commonsense knowledge about day-to-day events. The dataset specifies effects, needs, intents and attributes of the actors in an event. The average phrase length of nodes (4.40 words) is slightly higher than that of CN-100K. Multiple targets may exist for a source entity and relation. The tuples in this graph may also contain a `none` target in the case that the relation type does not necessitate an annotation. The original dataset split was created to make the set of seed entities between the training and evaluation splits mutually exclusive. Since the KB completion task requires entities to be seen at least once, we create a new random 80-10-10 split for the dataset. The development and test sets consisted of 87K tuples each.

[2]https://ttic.uchicago.edu/~kgimpel/commonsense.html
[3]https://homes.cs.washington.edu/~msap/atomic/

Dataset	# Nodes	# Edges	# Relations	Density	Average In-Degree
ConceptNet-100K	78088	100000	34	1.6e-5	1.25
ATOMIC	256570	610536	9	9.0e-6	2.25
FB15K-237 (for scale)	14505	272115	237	1.2e-3	16.98

Table 1: Knowledge Graph Statistics (Training Set Only). Graph density is calculated as $D = \frac{V}{N(N-1)}$, where N is the number of nodes and V is the number of edges in the graph.

3 Machine Commonsense Completion

We investigate two key ideas for performing completion of commonsense KGs – 1) transfer learning from language to knowledge graphs and 2) learning from graph structure. To address the challenge of sparsity of commonsense KGs, we enrich the graph connectivity with synthetic semantic similarity links to enable the effective use of GCNs. The overall architecture of our model is illustrated in Figure 4.

Problem Formulation

Given a knowledge graph $G = (N, V)$ where N is the set of nodes and V is the set of edges, we consider a single training instance as the tuple $v_i = (\mathbf{e}_1, \mathbf{rel}, \mathbf{e}_2)$ with source entity $\mathbf{e}_1$ represented in text as $\bar{\mathbf{e}_1}$, relation type $\mathbf{rel}$ and target entity $\mathbf{e}_2$, represented as $\bar{\mathbf{e}_2}$.[4] Here, $v_i \in V$ and $\mathbf{e}_1, \mathbf{e}_2 \in N$. The objective of the KB completion task is to maximize the score of a target entity $\mathbf{e}_2$ given a tuple prefix $(\mathbf{e}_1, \mathbf{rel})$. Following previous work (Dettmers et al. 2018), we also include inverse relations in our graph structure – for every edge $(\mathbf{e}_1, \mathbf{rel}, \mathbf{e}_2)$, we add an inverse edge $(\mathbf{e}_2, \mathbf{rel}^{-1}, \mathbf{e}_1)$.

Transfer Learning from Text to Knowledge Graphs

Transfer learning from language to knowledge graphs has recently been shown to be effective for commonsense knowledge graph construction (Bosselut et al. 2019). To transfer from language to knowledge graphs for completion, we finetune BERT (Devlin et al. 2019) with the masked language modeling loss and obtain rich semantic representations of nodes based on their text phrase. This allows BERT to be attuned to a KG's specific style of text. The input for finetuning is the list of unique phrases used to represent nodes in the KG. The format of the input to the model is `[CLS]` + $\bar{\mathbf{e}_i}$ + `[SEP]`, where $\bar{\mathbf{e}_i}$ is the natural language phrase represented by a node. We use representations of the `[CLS]` token from the last layer of the BERT model as node representations in our models. We represent the node embedding matrix obtained from the BERT model as $T \in \mathbb{R}^{|N| \times M}$, where M is the dimensionality of BERT embeddings.

Learning from Graph Structure

Graph Convolutional Networks (GCNs) (Kipf and Welling 2017) are effective at incorporating information from the local neighborhood of a node in the graph. A graph convolutional encoder take as input a graph G, and encodes each node as a D-dimensional embedding $h_i \in \mathbb{R}^D$ for all nodes $\mathbf{e}_i \in N$. The GCN encoder operates by sending

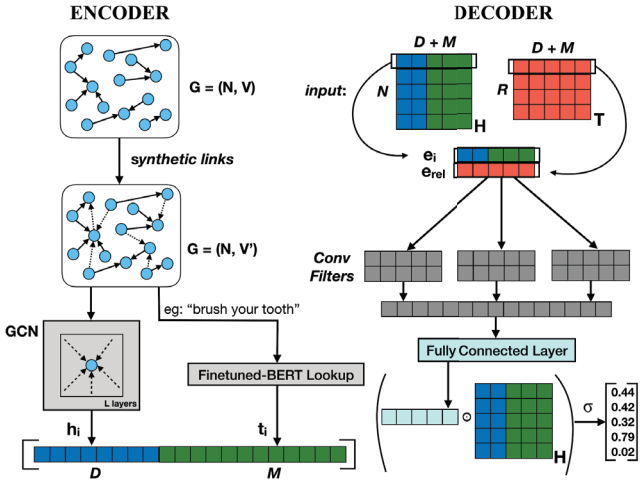

Figure 4: Model Architecture for machine commonsense completion.

messages from a node to its neighbors, optionally weighted by the relation type specified by the edge. This operation occurs in multiple layers, incorporating information multiple hops away from a node. The last layer's representation is used as the *graph embedding* of the node. Several variants (Schlichtkrull et al. 2018; Veličković et al. 2018) of these models have been proposed recently, all of which use the same underlying local neighborhood aggregation mechanism. We choose to use a version of the GCN which allows us to 1) parameterize the relation type corresponding to an edge and 2) account for the importance of a node's neighbor during aggregation. Given the graph G with R relation types and a GCN with L layers, the operation for computing the node representation of a node $\mathbf{e}_i$ in layer $l+1$ is:

$$h_i^{l+1} = \tanh\left(\sum_{r \in R}\sum_{j \in J_i} \alpha_r \beta_{ij}^l W^l h_j^l + W_0^l h_i^l\right) \quad (1)$$

where J_i represents the neighbors of the node $\mathbf{e}_i$ in the graph, and W^l is a linear projection matrix specific to layer l. The initial node representation h_i^0 is computed using an embedding layer. The second term in Equation 1 represents the self-connection for the node, and is used to propagate information from one layer to the next. α_r is the weight for the relation type of the edge and β_i^l is a vector denoting the relative importance of each of $\mathbf{e}_i$'s neighbors:

$$\beta_i^l = \text{softmax}(\hat{\boldsymbol{\beta}}_i^l) \quad (2)$$

[4] we use the terms tuple and edge synonymously.

where each element of $\hat{\boldsymbol{\beta}}_i^l$ is computed as,

$$\hat{\boldsymbol{\beta}}_{ij}^l = h_i^l h_j^l \tag{3}$$

Here, h_i^l and h_j^l are the representation of a node $\mathbf{e}_i$ and its neighbor $\mathbf{e}_j$. The output of the GCN is a node embedding matrix $H \in \mathbb{R}^{|N| \times D}$.

Graph Densification The sparsity of commonsense KGs makes it challenging for GCNs to perform information propagation over a node's neighborhood. To tackle this issue, we add synthetic edges between nodes with semantically similar meanings to boost the learning of graph embeddings. These edges form a new synthetic `sim` relation and are only used for computing graph embeddings but not scored by the decoder. To form these edges, we use the fine-tuned BERT model described earlier to extract node representations and use these representations to compute the cosine similarity between all pairs of nodes in the graph.

Upon computing the pairwise similarities, we use a hard threshold τ to filter the pairs of nodes that are most similar. This threshold is computed using different criteria for each graph, each prioritizing the precision of these links. For CN-100K ($\tau = 0.95$ results in 122,618 `sim` edges), we plot the distribution of pairwise similarity values between all pairs of nodes and select the top $\boldsymbol{\sigma}/2$ pairs of nodes to form these synthetic links, where $\boldsymbol{\sigma}$ is the standard deviation of the normally-distributed pairwise similarity distribution. For ATOMIC ($\tau = 0.98$ results in 89,682 `sim` edges), we obtain a pairwise similarity distribution that is not normal, and hence use a threshold (measured up to 2 decimal points) that would only increase up to 100K edges in the graph[5]. After this step, we obtain a set of edges V', where $|V'| > |V|$.

Progressive Masking for Fusion

Models that use node embeddings from GCNs and BERT tend to overly rely on BERT embeddings, rendering graph embeddings ineffective (we verify this using a random permutation test (Fisher, Rudin, and Dominici 2018) where we randomly shuffle graph embeddings in a minibatch and observe little drop in performance). At the beginning of training, graph embeddings are not informative whereas fine-tuned BERT embeddings provide useful information – causing the model to safely ignore graph embeddings. To prevent this issue, we randomly mask BERT embeddings starting with an all-zeros mask at the beginning to an all-ones mask at the end of 100 epochs[6]. The ratio of dimensions masked is set as (epoch/100) for the first 100 epochs. This strategy forces the model to rely on both sources of information. A similar technique was used to enforce multimodal machine translation models to rely on images by masking out tokens in the source (Caglayan et al. 2019).

[5]There are possibly other ways to choose the similarity threshold from a non-normal distribution, but we choose this criterion for the sake of simplicity. Decreasing this threshold resulted in minute drops in performance.

[6]midway while training for 200 epochs.

Convolutional Decoder

Convolutional models provide strong scores for KB completion (Dettmers et al. 2018; Shang et al. 2019) and hence, we use a convolutional decoder. Given an edge prefix $(\mathbf{e}_i, \mathbf{rel})$, graph embeddings $H \in \mathbb{R}^{|N| \times D}$ from the GCN (§3) and BERT-based node embeddings $T \in \mathbb{R}^{|N| \times M}$ (§3), the decoder produces a score for $(\mathbf{e}_i, \mathbf{rel}, \mathbf{e}_j)$ where $\mathbf{e}_i, \mathbf{e}_j \in N$. We use the convolutional decoder ConvTransE (Shang et al. 2019), to score a tuple. This model is based on ConvE (Dettmers et al. 2018) but additionally models the translational property of TransE (Bordes et al. 2013).

The model uses one of the following as input node embeddings – (i) graph embeddings $e_i = h_i$, (ii) BERT-based node embeddings $e_i = t_i$, or (iii) a concatenation of both $e_i = [h_i; t_i]$.[7] The model proceeds by first stacking the source node embeddings e_i, and relation embeddings e_{rel}, which are randomly initialized. The relation embeddings are chosen to have the same dimensionality as the embedding dimension of e_i, so that the stacking operation is possible. Assuming C different kernels and K as the width of a kernel, the output of kernel c is given by,

$$\begin{aligned} m_c(\mathbf{e}_i, \mathbf{rel})[\eta] = \sum_{\tau=0}^{K-1} & W_c(\tau, 0) e_i(\eta + \tau) \\ & + W_c(\tau, 1) e_{rel}(\eta + \tau) \end{aligned} \tag{4}$$

We denote the output for the kernel c to be $M_c \in \mathbb{R}^{|e_i|}$ and the concatenated output for all kernels to be $M \in \mathbb{R}^{C|e_i|}$. Finally, the score for a tuple $(\mathbf{e}_i, \mathbf{rel}, \mathbf{e}_j)$ is computed as,

$$s(\mathbf{e}_i, \mathbf{rel}, \mathbf{e}_j) = \sigma(M(e_i, e_{rel}) W_{conv} e_j) \tag{5}$$

where $W_{conv} \in \mathbb{R}^{C|e_i| \times |e_i|}$ is a bilinear projection matrix and σ is the sigmoid function. Upon computing scores for all candidate tuples $s(\mathbf{e}_i, \mathbf{rel}, \mathbf{e}_j)$, we use a binary cross entropy loss to train the model. All target nodes found in the training set are treated as positive instances while all non-existing target nodes are treated as negative instances.

Subgraph Sampling

As the graph size increases, it becomes computationally intensive to train with the entire graph in memory. Specifically, it is intensive to perform graph convolution over the entire graph and compute scores for all nodes in the graph using the decoder. For instance, the model with GCN and BERT representations for ATOMIC occupies ∼30GB memory and takes 8-10 days for training on a Quadro RTX 8000 GPU. Hence, we sample a smaller subgraph for training. We experiment with different sampling criteria and find that sampling edges uniformly at random provides the best performance.[8] For graph densification, we link all pairs of nodes in the subgraph that cross the semantic similarity threshold τ.

[7]We experimented with other fusion methods (e.g., summation, linear transformation of concatenated representations, initializing GCN with BERT embeddings, tying initial graph embeddings with BERT embeddings), and found that concatenation as a fusion method works best.

[8]We tried using random walk sampling, snowball sampling and uniform random sampling.

4 Experimental Setup

Evaluation Metrics

Following previous work on KB completion (Yang et al. 2015; Dettmers et al. 2018), we use ranking metrics (HITS and Mean Reciprocal Rank) for the purpose of evaluation. Similar to Dettmers et al. (2018), we filter out all remaining entities valid for an $(\mathbf{e}_1, \mathbf{rel})$ pair (found across training + validation + test sets) from the ranking when computing scores for a gold target entity. The scores are measured in both directions, where we compute the ranking of $\mathbf{e}_2$ given $(\mathbf{e}_1, \mathbf{rel})$ and the ranking of $\mathbf{e}_1$ given $(\mathbf{e}_2, \mathbf{rel}^-)$. We report the mean HITS and MRR scores averaged across both directions. We note that there are problems with these automatic metrics. Since commonsense KGs are highly sparse, several false negative target entities appear at the top of the ranking. Hence, we also perform human evaluation.

Baselines

We compare several high performing KB completion models and a transformer-based commonsense generation model.

DistMult DistMult, proposed by Yang et al. (2015), is an embedding-based method based on a bi-linear product between the entities and a relation matrix, which is constrained to be a diagonal matrix. The score function for a tuple is formulated as $s(\mathbf{e}_1, \mathbf{rel}, \mathbf{e}_2) = e_1^T W_{\mathbf{rel}} e_2$.

ComplEx Trouillon et al. (2016) proposed the use of complex-valued embeddings for nodes and relations, with the motivation that composition of complex embeddings can allow the model to handle a large number of relations. The model uses a tri-linear dot product as the scoring function.

ConvE ConvE (Dettmers et al. 2018) is a convolution-based model that stacks the node embedding and relation embedding for an entity prefix $(\mathbf{e}_1, \mathbf{rel})$ and reshapes the resulting tensor. The model performs 2D convolution upon this reshaped tensor and projects the output to a vector with the same dimensionality as the node embeddings.

ConvTransE Shang et al. (2019) proposed the ConvTransE model that builds upon the ConvE model but additionally models the translational properties of TransE. This model is discussed in depth in Section 3.

COMeT We adapt a commonsense generation model COMeT (Bosselut et al. 2019) for completion. COMeT generates the target phrase for a given (source entity, relation) prefix. Since COMeT was not trained using inverse relations, we only compute scores in the forward direction. We use the COMeT model by ranking target nodes based on the total and normalized negative log-likelihood score of each candidate tuple. In the absence of a standard approach for calibrating generation log-likelihoods, we report results using both metrics.[9] The COMeT scores for ATOMIC have been computed on a smaller evaluation set with 2000 tuples due to computational limitations (denoted by * in table).

Proposed Models

All our proposed models use the ConvTransE decoder. We experiment with using a GCN as the encoder for obtaining node embeddings, these models are labeled with the prefix GCN +. Models that utilize synthetic links in the graph are labeled with the affix SIM +. For models enriched with BERT representations, the ConvTransE decoder takes phrase representations extracted from BERT (BERT +) or a concatenation of graph embeddings and BERT embeddings (GCN + BERT +) as input node embeddings.

Training Regimen

We train all models for at least 200 epochs and continue training until the MRR on the development set stops improving. The MRR is evaluated on the dev set every 10 epochs for CN-100K and every 30 epochs for ATOMIC, and the model checkpoint with the highest MRR is used for testing. Further details about hyperparameter settings are specified in the supplemental material.

5 Results and Discussion

KB Completion Performance

We report our results on completion with subgraph sampling in Table 2. For completeness, we also report results for CN-100K with the full graph in Table 3. The baseline models are trained with the full graph in memory. We attempt to answer some pertinent research questions based on our results.

What does BERT know about commonsense assertions?

Main Finding: *BERT is proficient at capturing taxonomic relations and hence, provides significant boosts for CN-100K but is not as effective for ATOMIC.*

When BERT representations are incorporated into a model, performance improvements ranging from ∼19-35 points on MRR are observed for CN-100K. This indicates that BERT can provide crucial information for the model to discriminate between target entities. We discuss the reasons for the usefulness of BERT representations below.

Assertions of the style present in CN-100K are prevalent in large text corpora, which indicates that masking of tokens during pre-training would have enabled BERT to already gain a headstart on the task of KB completion. For instance, consider a sentence "*John bought plastic cups for the party.*" where we mask one token to obtain "*John bought* [MASK] *cups for the party.*" during pre-training. Now when the BERT representations are used to score a candidate tuple [cups, MadeOf, plastic], the model might compute a high score for this tuple because the BERT model solved a similar task during pre-training due to the masked LM objective. This result concurs with recent investigations on

[9] The total score can be biased against longer sequences; the normalized score can be biased against shorter sequences.

	CN-100K				ATOMIC			
	MRR	HITS@1	@3	@10	MRR	HITS@1	@3	@10
DISTMULT	8.97	4.51	9.76	17.44	12.39	9.24	15.18	18.30
COMPLEX	11.40	7.42	12.45	19.01	**14.24**	**13.27**	14.13	15.96
CONVE	20.88	13.97	22.91	34.02	10.07	8.24	10.29	13.37
CONVTRANSE	18.68	7.87	23.87	38.95	12.94	12.92	12.95	12.98
COMET-NORMALIZED	6.07	0.08	2.92	21.17	3.36*	0.00*	2.15*	15.75*
COMET-TOTAL	6.21	0.00	0.00	24.00	4.91*	0.00*	2.40*	21.60*
BERT + CONVTRANSE	49.56	38.12	55.5	71.54	12.33	10.21	12.78	16.20
GCN + CONVTRANSE	29.80	21.25	33.04	47.50	13.12	10.70	13.74	17.68
SIM + GCN + CONVTRANSE	30.03	21.33	33.46	46.75	13.88	11.50	**14.44**	**18.38**
GCN + BERT + CONVTRANSE	50.38	38.79	56.46	72.96	10.8	9.04	11.21	14.10
SIM + GCN + BERT + CONVTRANSE	**51.11**	**39.42**	**59.58**	**73.59**	10.33	8.41	10.79	13.86

Table 2: KB Completion Results on CN-100K and ATOMIC with **subgraph sampling**. We present baselines in the top half of the graph and our implementations in the bottom half. '*' indicates difference in evaluation set described in Section 4.

	CN-100K			
	MRR	HITS@1	@3	@10
BERT + CONV..	52.25	41.04	58.46	73.50
GCN + CONV..	27.24	18.96	30.46	43.17
SIM + GCN + CONV..	27.51	19.04	30.79	45.46
GCN + BERT + CONV..	50.80	39.29	57.33	72.66
SIM + GCN + BERT + CONV..	50.17	38.71	57.08	72.00

Table 3: KB Completion Results on CN-100K with **full graph training**. CONV.. is ConvTransE.

retrieval of commonsense knowledge from language models (Petroni et al. 2019; Feldman, Davison, and Rush 2019). Interestingly, BERT representations are found to be less useful for ATOMIC, because of a significantly larger number of nodes and more complex relation types (e.g. `oEffect` – effect of event on others, `xNeed` – what actor might need to do before event) in ATOMIC.

Finally, we attempt to use link prediction as a scaffold to find the commonsense relations that BERT captures well. Specifically, we use the BERT+CONVTRANSE model trained for CN-100K to find the top scoring relations (by rank) in the evaluation set (presented in Supp:Figure 8). We observe that BERT is most proficient at picking up on taxonomic relations such as `MadeOf`, `PartOf` and additionally, also does well on temporal relations such as `HasPrerequisite` and `ReceivesAction`. This can be explained by the higher frequency of taxonomic knowledge present in large text corpora as opposed to more complicated relations found in ATOMIC.

How crucial is graph structure information?

Main Finding: *Local graph structure information boosts performance when training with subgraph sampling, but improvements fade when training with the entire graph.*

We note that in the absence of BERT representations, incorporating graph embeddings provides strong improvements ($>$ +9 points on MRR for CN-100K and +$\sim$0.2 points for ATOMIC) over the model without a GCN encoder (i.e. GCN+CONVTRANSE is better than CONVTRANSE). Similarly in the presence of BERT representations, the graph embeddings provide complementary information about the local neighborhood of a node which boosts performance for CN-100K. When training with the entire graph in memory, the improvements with using graph embeddings fade away. However, it is important to note that, using graph embeddings provides benefits when it is infeasible to train models with the entire graph in memory.

We further verify the importance of graph embeddings when BERT-based representations are used with a random permutation test (Fisher, Rudin, and Dominici 2018). In this test, we randomly shuffle the graph embeddings within each minibatch during inference and note the drop in performance. For the SIM+GCN+BERT+CONVTRANSE model, we observe ΔMRR drops of -7.24, -8.98 for CN-100K and ATOMIC, respectively.

Do similarity-induced edges help?

Main Finding: *Similarity-induced edges boost learning of graph embeddings resulting in improved performance.*

We observe that when both graphs are trained with subgraph sampling but in the absence of BERT representations, augmenting the graph with similarity-induced edges provides improvements (SIM+GCN+CONVTRANSE $>$ GCN+CONVTRANSE). When BERT representations are incorporated, the similarity-induced edges continue to help for CN-100K but not for ATOMIC. The SIM+GCN+BERT+CONVTRANSE model achieves the best MRR for CN-100K. For ATOMIC, the SIM+GCN+CONVTRANSE model provides even stronger results than models which use BERT (+1.5 MRR points over BERT+CONVTRANSE). This allows us to conclude that augmenting commonsense KGs with similarity-induced links can provide more context for computing graph embeddings.

Can we use generative models for ranking tuples?

Main Finding: *Generative models cannot easily be repurposed to rank tuples for KB completion.*

Although generative models like COMET have shown to produce diverse and precise target phrases, our results from Table 2 indicate that it is non-trivial to repurpose them to perform completion. This is partly due to the problems

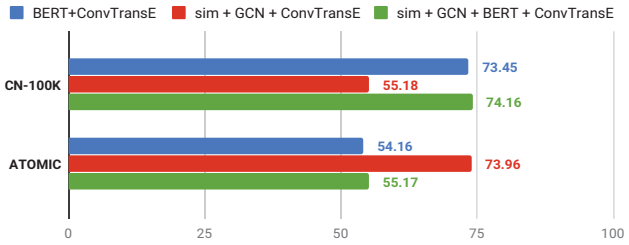

Figure 5: Human Evaluation Results (Average % of *valid* tuples among top-10 target candidates for random subset of entity-relation pairs)

associated with using log-likelihood as an estimate for the truth of a tuple. Nonetheless, generative models such as COMET have several merits. These models are faster to train, require lower memory for storage and are transductive in nature. However, we argue that reasoning models that rely on KBs could benefit more from a discriminative approach towards commonsense knowledge induction, one that would make the graph denser without adding new nodes.

Human Evaluation

Upon looking at model predictions, we note that false negative entities can appear at the top of the ranking. For instance, when $\mathbf{e}_1$=`PersonX wins two tickets` and **rel**=`xEffect` and the gold target entity $\mathbf{e}_2$=`elated and excited`, our model predicts `elated` and `excited` separately in the top 10 candidate entities. Automated metrics fail to capture this semantic similarity in predictions. Hence, we perform human evaluation by presenting the top 10 predicted targets to 3 annotators on Amazon Mechanical Turk (AMT), for a subset of the instances in the dev set. This subset was created by sampling 200 instances randomly for CN-100K, and 450 instances (=50 instances for 9 relations) for ATOMIC. The annotators were asked to answer whether a tuple is valid, where a valid tuple is meaningful and true. Our results are reported in Figure 5. Samples of our model predictions with human annotations are provided in the supplementary material. Human evaluation results indicate that using graph embeddings computed with graph densification in addition to BERT shows improvements.

6 Related Work

A host of techniques (Bordes et al. 2013; Yang et al. 2015; Trouillon et al. 2016; Dettmers et al. 2018; Shang et al. 2019) have been proposed for KB completion. These can be classified into graph traversal based and embedding-based approaches. Although embedding-based methods are more scalable, they usually assume enough training instances for each relation and a relatively high average degree for nodes to generalize well. Both these criteria are typically not satisfied by commonsense KGs.

Among embedding-based methods, convolutional models have proven useful for computing entity-relation feature representations (Dettmers et al. 2018; Shang et al. 2019) and hence, we use a convolutional decoder. Simultaneously, we make use of GCNs to incorporate a node's neighborhood into its representation. Much as we do, Shang et al. (2019) use a GCN to compute graph embeddings and the ConvTransE decoder to score tuples. Our work differs from their model, as we account for the relative importance of neighbors in our graph convolution operation, tackle the sparsity of the graph structure by graph densification, and provide strategies to jointly train using graph embeddings and BERT representations. Finally, the use of BERT embeddings follows from work which showed that initializing node embeddings with pre-trained GloVe vectors (Pennington, Socher, and Manning 2014) improves KB completion performance (Guu, Miller, and Liang 2015).

Prior work on completing commonsense KBs (Li et al. 2016; Saito et al. 2018; Jastrzebski et al. 2018) uses BiLSTMs to encode a tuple and performs linear transformations to predict a score for binary classification. We argue that commonsense KB completion models should be trained and evaluated for ranking. A similar effort aimed at predicting new knowledge in ConceptNet, using dimensionality reduction was made by Speer, Havasi, and Lieberman (2008).

We believe that completing commonsense KBs is bound to translate into improvements on a range of tasks that rely on these KBs, such as information retrieval (Kotov and Zhai 2012), question answering (Bauer, Wang, and Bansal 2018; Musa et al. 2018), and reading comprehension (Ostermann et al. 2018). Storks, Gao, and Chai (2019) provide a survey of the use of commonsense KBs in downstream tasks.

7 Conclusion

In this work, we show that existing KB completion models underperform with commonsense knowledge graphs, due to the sparsity and scale of these graphs. As our solution, we propose novel KB completion models, which are enriched by structural and semantic context, obtained from GCNs and language model representations. We describe a progressive masking strategy to efficiently utilize information from both sources. Further, we show that augmenting the graph with semantic similarity edges can help with completion. Our results indicate that 1) BERT-based node representations can provide significant improvements, especially when text in the graph is similar to pre-training corpora; 2) graph embeddings can provide rich local context for encoding nodes, and boost performance when training with subgraphs.

8 Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. This research was supported in part by NSF (IIS-1524371, IIS-1714566), DARPA under the CwC program through the ARO (W911NF-15-1-0543), and DARPA under the MCS program through NIWC Pacific (N66001-19-2-4031).

9 Supplementary Material

Hyperparameter Details

BERT Fine-tuning We used a maximum sequence length of 64, batch size of 32, and learning rate of 3e-5 to fine-tune

e1	rel	e2	Annotation
meet person	**HasPrerequisite**	*socialize*	**Y**
enjoy company of your friend	**HasSubevent**	*have fun*	**Y**
asteroid	**AtLocation**	*orbit*	**Y**
drunk	**AtLocation**	*motel*	**Y**
take examination	**Causes**	*headache*	**Y**
painter	**CapableOf**	*paint on canvas*	**Y**
dinner party	**HasProperty**	*fun*	**Y**
wallet	**MadeOf**	*money*	**N**
french horn	**IsA**	*musical instrument*	**Y**

Figure 6: Randomly sampled top-1 predictions from best model (SIM+GCN+CONVTRANSE+BERT) for CN-100K along with human annotations for validity of tuples.

e1	rel	e2	Annotation
PersonX lands new job	**xReact**	*happy*	**Y**
PersonX calls a taxi	**xNeed**	*to get in the car*	**N**
PersonX gives PersonY opoprtunities	**xAttr**	*helpful*	**Y**
PersonX moves PersonX's furniture	**xEffect**	*gets tired*	**Y**
PersonX drinks a lot	**xWant**	*to take a shower*	**N**
PersonX searches the entire house	**xIntent**	*to find something*	**Y**
PersonX sees PersonX's sister	**oReact**	*happy*	**Y**
PersonX buys a Christmas tree	**oEffect**	*none*	**Y**
PersonX rolls onto PersonY's back	**oWant**	*to have fun*	**Y**

Figure 7: Randomly sampled top-1 predictions from best model (SIM+GCN+CONVTRANSE) for ATOMIC along with human annotations for validity of tuples. While the complete relation descriptions can be found in Sap et al. (2019), it is noted that "x" refers to PersonX and "o" refers to other actors besides PersonX.

the uncased BERT-Large model with the masked language modeling objective. The warmup proportion was set to 0.1.

Baseline Models To train the baseline models, we used the implementations provided here.[10] We tuned the batch size and learning rate for the baseline models from 128, 256, 512 and 0.001, 0.002, 0.003 and used the default values for other hyperparameters.

[10]https://github.com/TimDettmers/ConvE

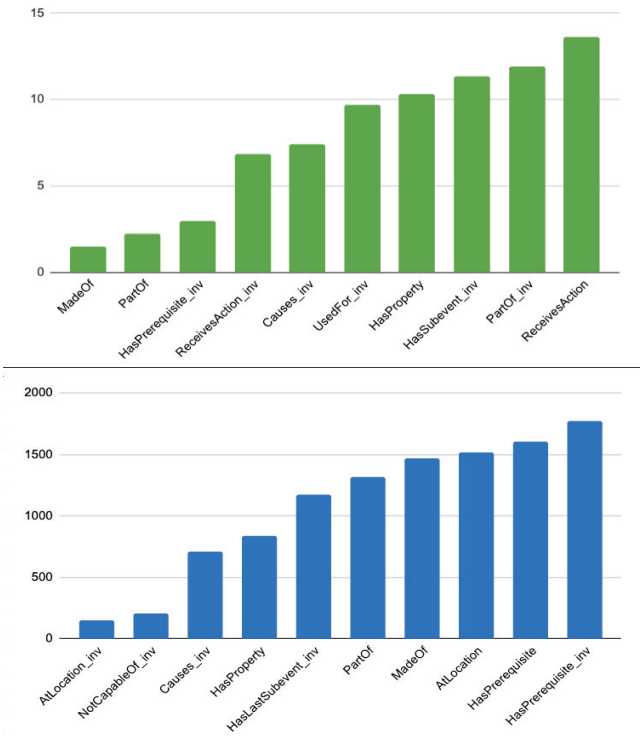

Figure 8: Top scoring relations for BERT+CONVTRANSE (top) and SIM+GCN+CONVTRANSE (bottom).

Our Implementations The graph convolutional network used 2 layers (using more ({3,4,5}) layers did not result in significant improvements) and an input and output embedding dimension of 200. The message passing algorithm for the GCN-based models was implemented using the Deep Graph Library (DGL). All embedding layers are initialized with Glorot initialization. The graph batch size used for subgraph sampling was 30000 edges. For the ConvTransE decoder, we used 500 channels, a kernel size of 5 and a batch size of 128. Dropout was enforced at the feature map layers, the input layer and after the fully connected layer in the decoder, with a value of 0.2. The Adam optimizer was used for optimization with a learning rate of 1e-4 and gradient clipping was performed with a max gradient norm value of 1.0. We performed L2 weight regularization with a weight of 0.1. We also used label smoothing with a value of 0.1.

Example Predictions

References

Bauer, L.; Wang, Y.; and Bansal, M. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4220–4230. ACL.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling

multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *The 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Caglayan, O.; Madhyastha, P.; Specia, L.; and Barrault, L. 2019. Probing the need for visual context in multimodal machine translation. In *NAACL 2019*. ACL.

Cambria, E.; Olsher, D.; and Rajagopal, D. 2014. Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In *Twenty-eighth AAAI conference on artificial intelligence*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL 2019*.

Feldman, J.; Davison, J.; and Rush, A. M. 2019. Commonsense knowledge mining from pretrained models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fisher, A.; Rudin, C.; and Dominici, F. 2018. Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective. *arXiv preprint arXiv:1801.01489*.

Guu, K.; Miller, J.; and Liang, P. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 318–327. Lisbon, Portugal: ACL.

Jastrzebski, S.; Bahdanau, D.; Hosseini, S.; Noukhovitch, M.; Bengio, Y.; and Cheung, J. 2018. Commonsense mining as knowledge base completion? a study on the impact of novelty. In *Proceedings of the Workshop on Generalization in the Age of Deep Learning*, 8–16. ACL.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Kotov, A., and Zhai, C. 2012. Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries. In *Proceedings of the fifth ACM international conference on Web search and data mining*, 403–412. ACM.

Lenat, D. B. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.

Li, X.; Taheri, A.; Tu, L.; and Gimpel, K. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1445–1455. ACL.

Musa, R.; Wang, X.; Fokoue, A.; Mattei, N.; Chang, M.; Kapanipathi, P.; Makni, B.; Talamadupula, K.; and Witbrock, M. 2018. Answering science exam questions using query reformulation with background knowledge.

Ostermann, S.; Roth, M.; Modi, A.; Thater, S.; and Pinkal, M. 2018. Semeval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, 747–757.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. ACL.

Petroni, F.; Rocktaschel, T.; Lewis, P.; Bakhtin, A.; Wu1, Y.; Miller, A. H.; and Riedel, S. 2019. Language models as knowledge bases? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Saito, I.; Nishida, K.; Asano, H.; and Tomita, J. 2018. Commonsense knowledge base completion and generation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 141–150. ACL.

Sap, M.; LeBras, R.; Allaway, E.; Bhagavatula, C.; Lourie, N.; Rashkin, H.; Roof, B.; Smith, N. A.; and Choi, Y. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *AAAI*.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; and Zhou, B. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*.

Speer, R., and Havasi, C. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*. Springer. 161–176.

Speer, R.; Havasi, C.; and Lieberman, H. 2008. Analogyspace: Reducing the dimensionality of common sense knowledge. In *AAAI*, volume 8, 548–553.

Storks, S.; Gao, Q.; and Chai, J. Y. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.

Toutanova, K., and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 57–66. ACL.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR)*.