UNDERSTANDING KNOWLEDGE INTEGRATION IN LANGUAGE MODELS WITH GRAPH CONVOLUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained language models (LMs) are not very good at robustly capturing factual knowledge. This has led to the development of a number of knowledge integration (KI) methods which aim to incorporate external knowledge into pretrained LMs. Even though KI methods show some performance gains over base LMs, the efficacy and limitations of these methods are not well-understood. For instance, it is unclear how and what kind of knowledge is effectively integrated into LMs and if such integration may lead to catastrophic forgetting of already learned knowledge. In this paper, we revisit the KI process from the view of graph signal processing and show that KI could be interpreted using a graph convolution operation. We propose a simple probe model called *Graph Convolution Simulator* (GCS) for interpreting knowledge-enhanced LMs and exposing what kind of knowledge is integrated into these models. We conduct experiments to verify that our GCS model can indeed be used to correctly interpret the KI process, and we use it to analyze two typical knowledge-enhanced LMs: K-Adapter and ERNIE. We find that only a small amount of factual knowledge is captured in these models during integration. While K-Adapter is better at integrating simple relational knowledge, complex relational knowledge is integrated better in ERNIE. We further find that while K-Adapter struggles to integrate time-related knowledge, it successfully integrates knowledge of unpopular entities and relations. Our analysis also show some challenges in KI. In particular, we find simply increasing the size of the KI corpus may not lead to better KI and more fundamental advances may be needed.

1 INTRODUCTION

Pretrained language models (LMs) have achieved state-of-the-art performance across various natural language processing (NLP) tasks. Previous works have shown that linguistic knowledge is captured quite well by LMs and it plays a vital role in their success (Liu et al., 2019a; Jawahar et al., 2019). However, factual knowledge is sparse and is expressed in varied ways in text. Thus, LMs are much worse in capturing factual knowledge about the world (Petroni et al., 2019; Wang et al., 2021b). This has led to the development of a variety of knowledge integration (KI) methods which aim of integrate external knowledge into LMs (Colon-Hernandez et al., 2021; Wang et al., 2021a; Zhang et al., 2019). Even though knowledge-enhanced LMs perform better on knowledge-related tasks, we lack a deep understanding about the inner workings of these models. Better downstream performance indicates that some new knowledge has been integrated, but how much knowledge has been successfully integrated, which type of knowledge is integrated is not well-understood.

Probing methods have been developed to understand the kinds of knowledge encoded in LMs (Petroni et al., 2019; Hewitt & Manning, 2019; Pimentel et al., 2020; Hou & Sachan, 2021). However, these methods cannot be easily used to understand the KI process. First, there are a large number of approaches for KI. KI in LMs can be implemented by matching sentences to entities or triples in knowledge graphs – called entity-wise integration (Peters et al., 2019; Zhang et al., 2019) and triple-wise integration (Liu et al., 2020; Wang et al., 2021a). There are several modeling choices (Colon-Hernandez et al., 2021) for KI, including proposing some modifications to the Transformer architecture (Peters et al., 2019; Zhang et al., 2019; Liu et al., 2020), verbalizing knowledge triples and using data augmentation for finetuning (Agarwal et al., 2021), and designing objective functions that predict the factual knowledge (Yao et al., 2019; Wang et al., 2021a). Thus, understanding the KI process requires building probes separately for each of these KI methods.

Furthermore, KI is typically implemented in a continual learning setup (Parisi et al., 2019) – KI is usually a secondary pretraining or finetuning step (Lu et al., 2021). As new knowledge is integrated, old knowledge could be catastrophically forgotten (CF; Kirkpatrick et al., 2016). KI could also lead to a situation called catastrophic remembering (CR; Kaushik et al., 2021), where the old knowledge may prevent the integration of new knowledge. Our understanding of these issues is limited.

Thus, in this paper, we propose a general method for understanding KI in pretrained LMs (§2). The method can be used to understand KI in knowledge-enhanced LMs trained in different ways. We use mutual information (MI) between the integrated knowledge organized as knowledge graphs (KGs) and text representations to estimate the amount of knowledge that has been encoded in the LM. To understand the continual KI process, integration of new knowledge and the forgetting of old knowledge is measured by the change in MI values as knowledge is integrated into the base LM (§2.1). However, MI is notoriously difficult to estimate. Besides, change in MI can only expose us to a macro view of KI, i.e., it will tell us how much knowledge is integrated. We are also interested in understanding which (type of) knowledge is integrated or forgotten. To address this, we revisit the KI process from the graph signal processing view, and show that the KI process can be simulated by the Graph Fourier transformation (§2.2), which can be simplified using graph convolutions (§2.3). Finally, the KI process can be interpreted using attention coefficients in the graph attention. The attention coefficients can be used to understand which kind of knowledge (knowledge of which entities or triples) has been integrated, catastrophically remembered or forgotten.

We show that our designed probe model GCS can correctly simulate and interpret the KI process for two popular knowledge-enhanced LMs: K-Adapter (Wang et al., 2021a) and ERNIE (Zhang et al., 2019). We find that K-Adapter integrates simple relational knowledge well, while ERNIE is better at integrating complex relational knowledge. In our case study, we find that K-Adapter does not learn temporal knowledge at all. We further break down our analysis of KI in terms of type of relations and the popularity of entities. We find that catastrophic remembering often happens to simple relational knowledge (i.e. simple relational knowledge is harder to edit), while complex relational knowledge is often catastrophically forgotten. We also find that catastrophic forgetting easily happens to popular entities, while catastrophic remembering often happens when the entities are not very common. Finally, we investigate the correlation between the size of the KI corpus and KI quality. We find that there is no apparent positive relationship between them, suggesting that merely building larger KI datasets may not be enough and we may need to make more fundamental advances in KI.

2 Revisiting Knowledge Integration

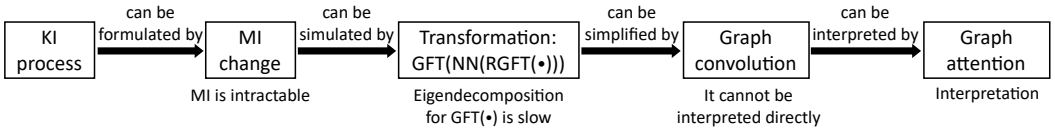


Figure 1: The various steps involved in deriving our probe model.

In this section, we describe the various steps involved in our method to interpret the KI process by the graph attention mechanism as shown in Figure 1. First, we formulate the KI process using MI. We use MI to measure the knowledge captured in LMs, and use the change in MI to understand KI, catastrophic remembering (CR), and catastrophic forgetting (CF). As MI is difficult to estimate, we simulate the KI process by a transformation. It is composed of the graph Fourier transformation (GFT), the inverse GFT (RGTF), and a neural network. However, the transformation is also not interpretable and GFT needs an explicit eigendecomposition of the Laplacian matrix of knowledge graphs (KGs) which is computationally prohibitive for large KGs. Thus, we circumvent this issue and simplify the transformation by using graph convolutions with well-approximated filters. The KI process then can be simulated by graph convolutions and interpreted by the graph attention mechanism. The notations used in this paper, together with their descriptions, are listed in Appendix A.

2.1 Knowledge Integration Definition

Before presenting a formal definition of KI based on MI, we introduce some basic concepts.

Knowledge graphs. We assume that factual knowledge can be formulated as a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $v_i \in \mathcal{V}$ represent entities, and edges in $\mathcal{E}$ represent relations between them. Let $\mathcal{N}_{v_i}$ denote the set of neighbors of node v_i, and t_i denote the entity text corresponding to the node v_i. Further, let $\boldsymbol{x}_i = \mathrm{LM}(t_i)$ denote the entity (text) representations of t_i given by a LM[1]. Let $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denote a matrix formed by stacking all entity representations $\boldsymbol{x}_i \in \mathbb{R}^d$. In this paper, we only consider nodes and relations in the KG and ignore other kinds of KG information such as edge weights, edge directions and multi-edges (multi-relations).

Knowledge probing. We present an information-theoretic probe (Hou & Sachan, 2021) which uses MI to measure the amount of knowledge integrated into LMs. We assume that the local graph structure $\mathcal{G}(v_i)$ contains all the factual information regarding v_i. In this work, we consider factual knowledge in the form of triples (v_i, r, v_j). Thus, it suffices to set $\mathcal{G}(v_i) = \mathcal{N}_{v_i}$. Factual knowledge that has been successfully learned should be reflected in the LM's entity representations. Let $\mathbf{x}$ be a random variable that takes values ranging over all possible entity representations $\boldsymbol{x}_i$ given by a LM[2], and $\mathbf{g}$ be a random variable that ranges over all possible corresponding local structures $\mathcal{G}(v_i)$. Intuitively, $\mathrm{MI}(\mathbf{x}; \mathbf{g})$ measures the amount of information in $\mathbf{g}$ that is contained in $\mathbf{x}$.

Knowledge Integration Definition. Given a KG $\mathcal{G}$, let the entity representation matrices given by a LM before and after KI be $\boldsymbol{X}$ and $\boldsymbol{H}$. The corresponding random variables are $\mathbf{g}$, $\mathbf{x}$, and $\mathbf{h}$. We can formulate KI as the change in MI: $\mathrm{MI}(\mathbf{x}; \mathbf{g}) \rightarrow \mathrm{MI}(\mathbf{h}; \mathbf{g})$ as knowledge of $\mathbf{g}$ is integrated into $\mathbf{x}$ to get $\mathbf{h}$. If $\mathrm{MI}(\mathbf{h}; \mathbf{g}) - \mathrm{MI}(\mathbf{x}; \mathbf{g}) \approx 0$, we say that catastrophic remembering (CR) has happened, i.e., there is a failure to integrate new knowledge. If catastrophic forgetting (CF) has happened, we have $\mathrm{MI}(\mathbf{h}; \mathbf{x}) \approx 0$.

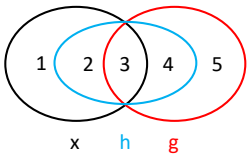


Figure 2: Venn diagram for MI. $\mathbf{x}$, $\mathbf{h}$, and $\mathbf{g}$ are random variables.

The definition can be intuitively visualized by Figure 2. The MI change can be represented by regions 1 and 4 in the Venn diagram. Ideally, if we have $\mathrm{MI}(\mathbf{h}; \mathbf{g}) - \mathrm{MI}(\mathbf{x}; \mathbf{g}) \approx \mathrm{MI}(\mathbf{g}; \mathbf{g}) - \mathrm{MI}(\mathbf{x}; \mathbf{g})$, i.e., the region 5 is small, we say most knowledge is successfully integrated. If little new knowledge has been integrated, the region 4 is very small. Then, we say that CR has happened. If CF happens, most knowledge in $\mathbf{x}$ is forgotten in $\mathbf{h}$ after KI, and region 1 is large. Successful KI happens when much new knowledge is integrated (i.e., $\mathrm{MI}(\mathbf{h}; \mathbf{g})$ is large) and little old knowledge is forgotten (i.e., $\mathrm{MI}(\mathbf{h}; \mathbf{x})$ is large).

Even if we can formally describe the KI process and CR and CF based on the MI, it is not easy to interpret KI directly using MI. Besides, only knowing MI values is not sufficient to answer questions such as which (type) of knowledge is integrated. Therefore, instead of calculating MI directly, we construct a transformation to simulate the KI process, which lends it to better interpretation.

2.2 Knowledge Integration Simulation

In this subsection, we show that we can construct a transformation that can approximate the MI change with arbitrary accuracy. We begin by introducing the concept of Graph Fourier transforms.

Graph Fourier transformation. Graph Fourier transform (GFT) is an important tool in spectral graph theory. GFT can be used to transform the entity representation matrix $\boldsymbol{X}$ in the Euclidean space to the graph space (i.e., KG space). Specifically, let $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ be the symmetric adjacency matrix corresponding to $\mathcal{G}$. Let $\boldsymbol{L}_n = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ denote the normalized Laplacian matrix for $\mathcal{G}$, where $\boldsymbol{D}$ denotes the degree matrix of $\mathcal{G}$. We do the eigendecomposition for $\boldsymbol{L}_n$ as $\boldsymbol{L}_n = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, where $\boldsymbol{U}$ is the matrix of eigenvectors ordered by eigenvalues and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ is the diagonal matrix of eigenvalues. Based on the GFT (Sandryhaila & Moura, 2014), formally, the transformation of $\boldsymbol{X}$ to the KG space can be written as $\mathrm{GFT}(\boldsymbol{X}) = \boldsymbol{U}^T\boldsymbol{X}$, and its inverse transformation can be written as $\mathrm{RGFT}(\mathrm{GFT}(\boldsymbol{X})) = \boldsymbol{U}\mathrm{GFT}(\boldsymbol{X}) = \boldsymbol{X}$. With GFT, we can transform entity representations to KG space and find a transformation there to simulate the KI process where $\mathbf{x}$ is changed to $\mathbf{h}$ by integrating information in $\mathbf{g}$,

Theorem 1 (Transformation Existence)**.** *Denote the graph Fourier transformation and its inverse transformation in terms of $\mathcal{G}$ as $\mathrm{GFT}(\cdot)$ and $\mathrm{RGFT}(\cdot)$. Given a LM and its knowledge-enhanced*

[1]We represent each entity as the average of its word(-piece) embeddings given by the LM as in Hewitt & Manning (2019) and Hou & Sachan (2021).

[2]Here, the set of entity representations $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ can be regarded as empirical samples from $\mathbf{x}$.

version, suppose that $\mathrm{MI}(\mathbf{h};\mathbf{g}) - \mathrm{MI}(\mathbf{x};\mathbf{g}) > 0$*, and there exists a mapping that satisfies* $f(\mathbf{x},\mathbf{g}) = \mathbf{h}$*. Then, for any* $\epsilon > 0$*, there exists a neural network* $\mathrm{NN}(\cdot)$ *such that*

$$|f(\mathbf{x},\mathbf{g}) - \mathrm{RGFT}(\mathrm{NN}(\mathrm{GFT}(\mathbf{x})))| < \epsilon. \tag{1}$$

The proof can be found in Appendix B. Theorem 1 shows that there exists an approximated transformation composed of GFT and a neural network to transform $\mathbf{x}$ and $\mathbf{g}$ to $\mathbf{h}$ with arbitrary accuracy. The transformation is a simulation of the KI process, i.e., MI change, and it promise the accuracy. However, it is difficult to interpret the KI process with the transformation. Besides, KGs are normally very large, and computing the eigendecomposition of the Laplacian matrix for GFT is prohibitively expensive. Therefore, to address these two issues, we take a deeper look into this transformation. We show that it can be simplified by graph convolutions, which is significantly faster and more interpretable.

2.3 Knowledge Integration Interpretation

In this subsection, we simplify the transformation in Theorem 1 and make it interpretable. To achieve this, we analyze the transformation and show that MI change only happens in linear functions of the neural network (Figure 3). We make use of the fact that linear functions in the graph space are equivalent to graph convolutions in the Euclidean space. Thus, we can eventually interpret KI by the graph attention mechanism in graph convolutions. We begin this subsection by introducing graph convolutions.

Graph convolution. Graph convolutions are often used to model relational information in KGs. In graph convolutions, entities aggregate information from their neighbors and pass this information along based on the graph. Graph convolutions can be implemented by filters g_Θ in the KG space. As the GFT of the convolution of g_Θ and $\boldsymbol{X}$ is the pointwise product of their GFT (Bracewell & Bracewell, 1986), the convolution can be written as $g_\Theta \star \boldsymbol{X} = \mathrm{RGFT}(g_\Theta \cdot \mathrm{GFT}(\boldsymbol{X}))$ (Bruna et al., 2014). To circumvent the expensive eigendecomposition in GFT, Defferrard et al. (2016) and Kipf & Welling (2017) propose well-approximated graph filters which work significantly faster. Velickovic et al. (2018) and Thekumparampil et al. (2018) introduce attention mechanism into graph filters, where the contribution of each edge to the convolution can be shown explicitly. The graph attention makes the filters more powerful and graph convolutions more interpretable (Fu et al., 2020).

Graph attention for KI interpretation. Suppose $\mathrm{MI}(\mathbf{x};\mathbf{g}) < \mathrm{MI}(\mathbf{h};\mathbf{g})$, $\mathrm{MI}(\mathbf{h};\mathbf{x}) < \mathrm{MI}(\mathbf{x};\mathbf{x})$, and the mapping $f(\mathbf{x},\mathbf{g}) = \mathbf{h}$ can be well approximated by a neural network with n layers. Considering that the graph convolution can be fast, and filters with attention mechanism are interpretable, we can use the graph attention to interpret KI, CR, and CF. The formal proof is in Appendix C.

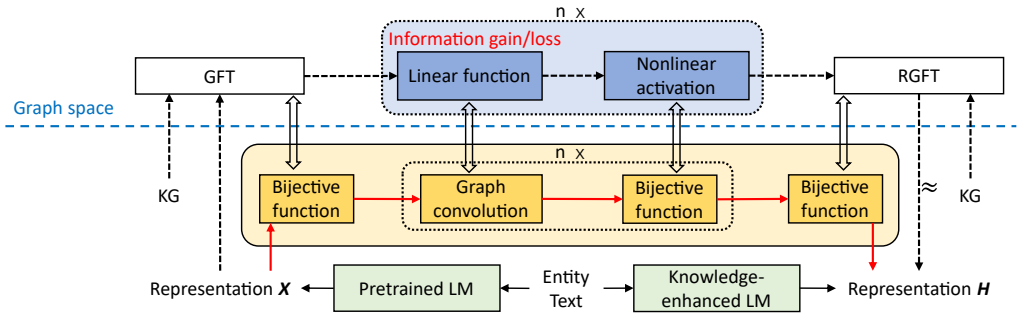


Figure 3: Illustration of the approximated transformation. Black dashed arrows show the approximated transformation for the KI process. Red arrows show the simplified transformation with graph convolutions. These two transformations are equivalent. GFT, RGFT, and nonlinear activation functions are bijective functions, and linear functions in the graph space are equivalent to graph convolutions.

Figure 3 briefly illustrates the overall idea. As proved in Theorem 1, the black dashed transformation can simulate the MI change with arbitrary accuracy. It is accurate, but slow and not interpretable. The red dashed transformation can be fast and interpretable. We show that both are equivalent. According to the invariance of MI (Kraskov et al., 2004), the introduction of bijective functions cannot change the MI. We know that GFT and RGFT are bijective. And considering that nonlinear activation functions in the neural network (e.g., $\mathrm{sigmoid}(\cdot)$ functions) are bijective as well, the

MI change in the KI process can only happen in the linear function. Based on the convolution theorem (Bracewell & Bracewell, 1986), linear functions in graph space are graph convolutions (Sandryhaila & Moura, 2014; Bruna et al., 2014; Kipf & Welling, 2017). Thus, we can use graph convolution operation instead to simulate the KI process in the graph space.

As for the eventual interpretation, Graph filters with attention mechanism are more powerful, and more interpretable. According to Fu et al. (2020), graph attention works as edge denoising. The contribution of each edge to the convolution can be exposed explicitly by the attention coefficient. It can also indicate the importance of neighbors, where neighbors that play an important role in the convolution have larger attention coefficients on connected edges. Thus, we use attention coefficients on edges to interpret which set of edges play an important role in the convolution, i.e., which set of triples are learned during KI. The attention mechanism can also be used to detect CF and CR situations. To do this, we add self-loops on entities in the KG. Attention coefficients on the self-loops indicate how much of the original information is retained by the corresponding entities. The amount of retained information can be used to indicate whether CF (if no information is retained) and CR (if all information is merely retained) has taken place.

3 Graph Convolution Simulator

In the previous section, we showed that the KI process can be simulated and interpreted by graph convolutions and the graph attention mechanism. Following that understanding, we design a Graph Convolution Simulator (GCS) model. We begin by introducing the GCS architecture and an objective to optimize it. Finally, we show how graph attention can be used to interpret the KI process.

GCS architecture. We design the GCS based on the simplified transformation. The GCS comprises of n graph convolutional layers and $n+2$ bijective functions, since the neural network is composed of n layers as shown in Figure 3 and Appendix B. As factual knowledge triples only include entities within one-hop neighborhoods, one graph convolutional layer is sufficient for entities to integrate knowledge from local structures ($n = 1$)[3]. We show that a composition of bijective functions is still bijective in Appendix D. $n+2$ bijective functions thus can be simplified to two bijective functions. To implement a bijective function in practice, we show that special MLP layers can be bijective if we add a small noise on the weight matrices. The formal description and the proofs are in Appendix D. Therefore, we design our GCS model with one graph convolutional layer and two bijective MLP layers as:

$$\mathrm{GCS}_{\theta_1}(\cdot) = \mathrm{MLP}_n(\mathrm{GC}(\mathrm{MLP}_n(\cdot), \mathcal{G})), \tag{2}$$

where $\mathrm{MLP}_n(\cdot)$ is the bijective MLP layer. And $\mathrm{GC}(\cdot, \mathcal{G})$ is the graph convolutional layer on the KG $\mathcal{G}$, which is used to simulate and interpret the KI process.

GCS objective. Consider that we define the KI process based on the MI change, i.e., $\mathrm{MI}(\mathbf{x}; \mathbf{g}) \rightarrow \mathrm{MI}(\mathbf{h}, \mathbf{g})$. To simulate how $\mathbf{x}$ is transformed to $\mathbf{h}$, the MI between the output of GCS and $\mathbf{h}$ should be large. Thus, we define the objective function by maximizing the MI as

$$\mathcal{L} = -\mathrm{MI}(\mathrm{GCS}_{\theta_1}(\mathbf{x}); \mathbf{h}). \tag{3}$$

We optimize MI equation 3 by maximizing the compression lemma lower bound (Banerjee, 2006) as in Belghazi et al. (2018). The inputs of GCS are $\boldsymbol{X}$, and let the output be denoted by $\boldsymbol{Z}$. We can regard $\boldsymbol{Z}$ and $\boldsymbol{H}$ as empirical samples of random variables $\mathbf{z}$ and $\mathbf{h}$. Thus, we have:

$$\mathrm{MI}(\mathbf{z}; \mathbf{h}) \geq \sup_{T \in \mathcal{F}} \mathbb{E}_{\mathbb{P}_{\mathbf{zh}}}[T] - \log(\mathbb{E}_{\mathbb{P}_{\mathbf{z}} \otimes \mathbb{P}_{\mathbf{h}}}[e^T]). \tag{4}$$

Here, $\mathcal{F}$ can be any class of functions $T : \Omega \rightarrow \mathbb{R}$ satisfying certain integrability constraints (Belghazi et al., 2018). $\mathbb{P}_{\mathbf{zh}}$ represents the joint distribution of $\mathbf{z}$ and $\mathbf{h}$, and $\mathbb{P}_{\mathbf{z}} \otimes \mathbb{P}_{\mathbf{h}}$ represents the product of their marginal distributions. In practice, we let $\mathcal{F} = \{T_{\theta_2}\}$ be the set of functions parameterized by a neural network, and optimize it by stochastic gradient descent. Then, the objective function can be rephrased as

$$\max_{\theta_1, \theta_2} \left(\mathbb{E}_{\mathbb{P}^{|\mathcal{V}|}_{\mathbf{z},\mathbf{h}}}[T_{\theta_2}] - \log\left(\mathbb{E}_{\mathbb{P}^{|\mathcal{V}|}_{\mathbf{z}} \otimes \mathbb{P}^{|\mathcal{V}|}_{\mathbf{h}}}[e^{T_{\theta_2}}] \right) \right), \text{ where } \mathbf{z} = \mathrm{GCS}_{\theta_1}(\mathbf{x}). \tag{5}$$

[3]Note that the number of graph convolutional layers decides the receptive field of entities. n layers represent that each entity can get information from its n-hop neighbors.

In equation 5, $\mathbb{P}_{\mathbf{z}}^{|\mathcal{V}|}$ represents the empirical distribution of $\mathbf{z}$, i.e., $\boldsymbol{Z}$. If the KG is very large, we can optimize the network by sampling a small subgraph of the KG.

Graph attention for interpretation. Given an entity v_i and its set of neighbors $\mathcal{N}_{v_i}$, we can write the graph convolutional layer as:

$$\mathrm{GC}(\boldsymbol{x}_i) = \sigma\left(\sum_{v_j \in \mathcal{N}_{v_i} \cup \{v_i\}} a_{i,j} \boldsymbol{W}^V \boldsymbol{x}_j\right), \text{ where } a_{i,j} = \mathrm{softmax}\left(\frac{(\boldsymbol{W}^Q \boldsymbol{x}_i) \cdot (\boldsymbol{W}^K \boldsymbol{x}_j)}{\sqrt{d_k} \cdot t}\right). \quad (6)$$

Here, $\boldsymbol{x}_i$ is the entity representation of v_i before knowledge integration, the activation function $\sigma(\cdot)$ is $\mathrm{ELU}(\cdot)$ function, and $\boldsymbol{W}^V$ is a weight matrix. $a_{i,j}$ is the attention coefficient on the edge that connects v_i and v_j. $\boldsymbol{W}^Q$ and $\boldsymbol{W}^K$ are two parameter matrices in the graph attention. d_k is the dimension of vector $\boldsymbol{W}^K \boldsymbol{x}_j$, and $\mathrm{softmax}(\cdot)$ is the edge-wise softmax function with respect to node v_i. Temperature t is a hyperparameter that controls the attention distribution to be hard or soft. As the multiplicative attention mechanism (Vaswani et al., 2017) is broadly used in LMs, we also select multiplicative attention in the graph attention.

As introduced in § 2.3, we use attention coefficients for interpretation. As for KI, large attention coefficients on edges mean that the corresponding knowledge triples are well learned, and vice versa. We add self-loops as in Equation 6, and use attention coefficients on self-loops for CR, and CF interpretation: a large value represents that the entity mainly focuses on itself, and there is no knowledge integrated, where CR happens to it. On the other hand, a small value indicates that the entity mainly focuses on its neighbors, where CF happens to it. Details of GCS are in Appendix F.

4 Experiments

We interpret the KI process by GCS for two typical knowledge-enhanced LMs: K-Adapter (Wang et al., 2021a) and ERNIE (Zhang et al., 2019). While K-Adapter uses triple-wise integration, ERNIE uses an entity-wise integration strategy. We first introduce details of K-Adapter and ERNIE. After that, we verify our GCS model, showing that GCS can correctly interpret the KI process. Then, we use GCS to analyze the KI process and analyze which types of relations/entities are (not) integrated during the KI process. Then, we classify knowledge in terms of the relation type, the relation popularity, and the entity popularity and also analyze how these methods fare on integrating different kinds of knowledge.

4.1 Knowledge-Enhanced Models

K-Adapter. Taking RoBERTa (Liu et al., 2019b) as the backbone model, K-Adapter inserts three new layers into RoBERTa (layer 0, 11, 22) to learn new knowledge. During the integration, parameters of RoBERTa are frozen, only parameters of the newly inserted layers are updated. K-Adapter uses the T-REx-rc (ElSahar et al., 2018) dataset which has an alignment of natural sentences with knowledge triples in Wikidata for KI. The model decides whether certain relations exist or not, and classifies relation labels given the aligned sentence. As knowledge is learned in newly inserted layers, the model no longer needs to use the T-REx-rc dataset, and the K-Adapter can be finetuned like any other pretrained LM on downstream tasks.

ERNIE. ERNIE integrates factual knowledge into BERT (Devlin et al., 2019) directly without introducing any extra parameters for KI. The Wikipedia corpus and Wikidata knowledge triples are selected for integration. As there is no provided alignment between natural sentences and knowledge entities, ERNIE uses TAGME (Ferragina & Scaiella, 2010) to extract entity mentions in sentences and aligns them with corresponding entities in KGs. A new objective is designed for KI in addition to the standard MLM and NSP objectives in BERT: alignments in the input text are randomly masked, and the model is asked to select aligned entities from KGs. Different from K-Adapter that stores factual knowledge in newly inserted layers, when ERNIE finds the aligned entity, its embedding obtained from Bordes et al. (2013) is integrated into the output representations. Thus, during inference, the KG and its embeddings are still required.

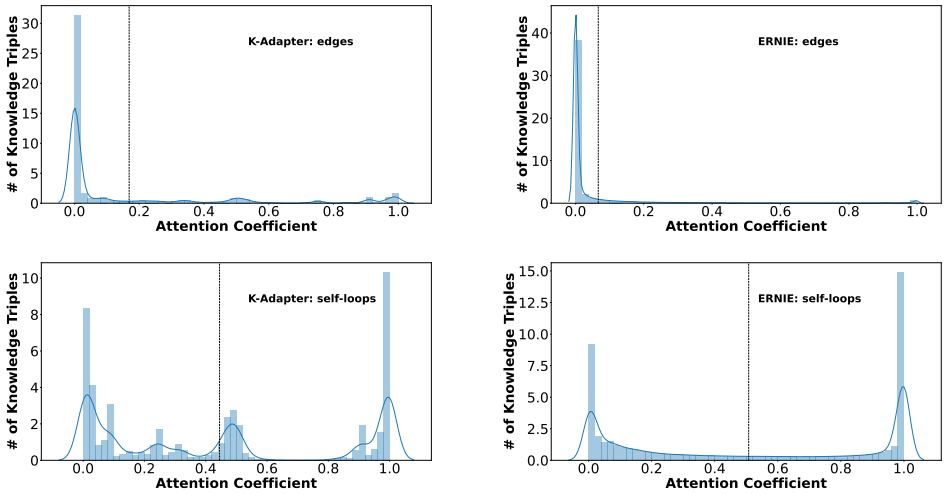


Figure 4: The attention coefficient distributions of edges and self-loops for K-Adapter and ERNIE. The histogram shows the empirical distributions (i.e., frequency), and the blue curves are the Gaussian kernel density estimate. The black dashed vertical lines indicate the average values.

4.2 Knowledge Integration Overview

Figure 4 presents the empirical distributions of attention coefficients for K-Adapter and ERNIE. The above two subfigures show distributions on edges, interpreting the KI from triples. While the below two subfigures show distributions of self-loops, illustrating the KI from entities[4]. We can find that most knowledge triples are not learned well for both K-Adapter and ERNIE (i.e., $a_{i,j} < 0.1$), while K-Adapter performs slightly better. When it comes to entity-wise integration, in general, entity knowledge is also not learned well. We find that both CR (i.e., $a_{i,i} > 0.9$) and CF (i.e., $a_{i,i} < 0.1$) happens for many entities, especially for ERNIE. K-Adapter outperforms ERNIE since some entities are integrated well (i.e., $0.4 < a_{i,i} < 0.6$)[5]. Note that in our following analysis, we simply regard triples with attention coefficients $a_{i,j} > 0.1$ on edges as learned triples. Entities with attention coefficients $0.4 < a_{i,i} < 0.6$ on self-loops as treated as well-learned entities. Entities with $a_{i,i} < 0.1$ on self-loops means CF has happened, and $a_{i,i} > 0.9$ means CR has happened.

4.3 GCS Verification

First, we reproduce the KI process of K-Adapter and ERNIE only using the learned knowledge (i.e., triples, and entities) identified by GCS[6]. Details of our hyperparameters are in Appendix E. We follow the experiments in K-Adapter and ERNIE, choosing entity typing as the downstream task, and two shared datasets: OpenEntity (Choi et al., 2018) and FIGER (Ling et al., 2015). We drop sentences aligned with knowledge triples with $a_{i,j} < 0.1$ for K-Adapter entities with $a_{i,i} > 0.9$ for

Table 1: Performance of K-Adapter, ERNIE, and their dropped versions on entity typing on the OpenEntity and FIGER datasets, respectively. The difference between knowledge-enhanced LMs and their dropped versions is not noticeable.

Model	OpenEntity					
	P	R	F1-Micro	P	R	F1-Macro
K-Adapter	76.63	75.26	75.94	62.97	62.37	62.67
K-Adapter (drop)	75.95 ↓	75.95 ↑	75.95 ↑	63.65 ↑	63.07 ↑	63.36 ↑
ERNIE	78.24	68.75	73.19	58.19	57.62	57.90
ERNIE (drop)	78.11 ↓	71.43 ↑	74.62 ↑	59.83 ↑	59.66 ↑	59.74 ↑
Model	**FIGER**					
	P	R	F1-Micro	P	R	F1-Macro
K-Adapter	67.50	88.79	76.69	74.07	89.00	80.85
K-Adapter (drop)	67.29 ↓	88.88 ↑	76.59 ↓	73.99 ↓	89.15 ↑	80.87 ↑
ERNIE	77.39	65.81	71.13	77.78	69.35	73.32
ERNIE (drop)	77.38 ↓	64.90 ↓	70.60 ↓	77.50 ↓	68.49 ↓	72.72 ↓

[4]For interpretation, attention coefficients on edges are used for triple-wise integration, and those on self-loops are for entity-wise integration.

[5]Note that users may choose different thresholds according to requirements. We heuristically set these thresholds for our analysis.

[6]Specifically, for K-Adapter and ERNIE, we reproduce them by retraining RoBERTa and BERT with their provided datasets. And for their dropped versions, we retrain RoBERTa and BERT with KI corpus aligned to learned knowledge (triples or entities) identified by GCS.

ERNIE and re-train K-Adapter and ERNIE on the remaining data. Table 1 presents the performance of entity typing. We find that dropping knowledge that GCS identifies as unlearned does not noticeably lower the performance of entity typing for both K-Adapter and ERNIE, which proves that GCS can correctly identify what kind of knowledge has been learned and what has not been learned yet. Besides, we further verify GCS by introducing a random dropping strategy as comparison. Detailed results can be found in Appendix H.

4.4 GCS FINDINGS

In this subsection, we present a microscopic view of the KI process. We classify knowledge based on relation types and popularity and interpret KI for different types of relations and on entities depending on their entity popularity. Implementation details for this experiment are in Appendix F.

Table 2 presents the KI, CR, CF analysis of K-Adapter and ERNIE for different types of relations: $1-1$ relations, $N-1$ relations, and $N-M$ relations. We find that for K-Adapter (that integrates knowledge in a triple-wise manner), knowledge of $N-M$ relations knowledge is not captured well. However, K-Adapter integrates knowledge of $1-1$ relations well. This phenomenon is common for Transformer encoders, where knowledge with complex structure cannot be captured well (Petroni et al., 2019). ERNIE (that uses knowledge graph embeddings for integration) shows different results. We find that entities connected with $1-1$ and $N-M$ relations are captured well. But for entities connected to $N-1$ relations, they are not integrated well. Since ERNIE relies on KG embedding to learn structured knowledge, KI is consistent with the quality of the KG embedding given by TransE (Bordes et al., 2013). Regarding CR and CF, we find that for both K-Adapter and ERNIE, CR happens more often for entities with simple structures, while CF is more common for entities in complex structures.

Table 2: Interpretation of KI, CR, and CF for K-Adapter and ERNIE in terms of different types of relation. We classify the relation into three types: $1-1$ relations, $N-1$ relations, and $N-M$ relations. We report ratios of learned knowledge (triples or entities). And we also report ratios of remembered and forgotten entities connected to specific types of relations.

Statistics \ Model	K-Adapter (Wang et al., 2021a) on T-REx-rc			
	$1-1$ relation	$N-1$ relation	$N-M$ relation	Total
# of triples	21,690	813,674	1,729,644	2,565,008
Learned triple ratio	58.89%	38.39%	24.00%	28.86%
# of connected entities	21,690	406,837	352,748	781,275
CR entity ratio	41.11%	31.72%	26.02%	29.41%
CF entity ratio	26.40%	30.29%	40.89%	34.97%
Statistics \ Model	**ERNIE (Zhang et al., 2019) on Wikidata**			
	$1-1$ relation	$N-1$ relation	$N-M$ relation	Total
# of connected entities	1,799	529,186	2,744,549	3,275,534
Learned entity ratio	70.65%	42.86%	73.33%	68.39%
CR entity ratio	29.41%	56.07%	26.67%	38.28%
CF entity ratio	23.18%	8.65%	37.10%	32.49%

Knowledge of different types of relations (*Wiki Count* and *Wiki data type*) is learned differently in K-Adapter (case study). In our next study, we select 6 relations aligned with roughly the same number of sentences from the T-REx-rc dataset shown in Table 3 and analyze how KI depends on the type and popularity of the relation. Appendix G shows the specific numbers. We report the *Count* and *Data type* properties in Wikidata[7] to indicate the popularity of relations, and their data types. We can find that even if {"*Place of birth*", "*Part of*"} has roughly the same *Count* as {"*Date of death*", "*Date of birth*"}. However, as the *Data type* of the latter set is "*Time*", knowledge of those set of relations cannot be learned by K-Adapter. We speculate that this is because Transformer encoders do not capture information about time well (Dhingra et al., 2021; Zhou et al., 2021). When comparing sets {"*Place of birth*", "*Part of*"} and {"*Located in the administrative territorial entity*", "*Country*"}, we can find that if relations have small *Wiki Count*, the knowledge triples are easier to capture during KI.

Table 3: The interpretation of KI for K-Adapter in terms of relations. We list 6 relations and classify them into three types based on the Wiki Count and Wiki data type. The ratio of learned knowledge triples are reported.

Relation label	T-REx-rc		
	Wiki Count	Wiki data type	Learned triple ratio
Place of birth	2,850,424	Wikibase item	10.95%
Part of	4,164,470	Wikibase item	17.25%
Date of death	2,637,358	Time	<0.01%
Date of birth	5,294,649	Time	<0.01%
Located in the administrative territorial entity	10,776,120	Wikibase item	6.13 %
Country	14,174,811	Wikibase item	0.12%
Total	-	-	10.09%

[7]https://www.wikidata.org/wiki/Wikidata:Database_reports/List_of_properties/all

Table 4: The interpretation of KI, CR, and CF for K-Adapter in terms of relations and entities. We list 3 typical relations: CR happens to most connected entities; CF happens to most connected entities; and KI, CR, CF happens equally to connected entities. And we list 5 correspondingly aligned sentences. The Google Ngram of entities are reported to show the popularity of entities.

Relation label	Ratio of triples connected to CR entities	Ratio of triples connected to CF entities	Ratio of triples connected to WL entities	# of aligned sentences
KI of Entities (Google Ngrams)	Examples			
Copyright license	85.09%	14.61%	0.16%	4,400
CR, $(4.95, 2.47)\times10^{-7}$	This article incorporates **public domain material** from the United States Geological Survey document "Wasson Rock" (content from the **Geographic Names Information System**).			
Office held by head of government	0.46%	99.54%	0.00%	14,315
CF, $(0.91, 1.33)\times10^{-2}$	Seven people served as governor of Colorado Territory over eight terms, appointed by the **President** of the **United States**.			
Opposite of	26.30%	41.87%	28.62%	11,577
CR, $(1.18, 1.77)\times10^{-4}$	It is well covered with **deciduous** and **evergreen** forests.			
CF, $(3.67, 5.60)\times10^{-2}$	The prison's north wing is filled with **left**-wing rebels while the south wing is filled with **right**-wing government supporters and paramilitaries.			
WL, $(6.83, 2.46)\times10^{-4}$	Wildlife of Iran includes its **flora** and **fauna** and their natural habitats.			

Knowledge of different entities (sorted by their popularity) has different CR and CF ratios (case study). Table 4 reports the KI for K-Adapter in terms of entities. We select three relations {"*Copyright license*", "*Office held by head of government*", "*Opposite of*"}. CR happens to most connected entities for relation "*Copyright license*". CF happens to most connected entities for relation "*Office held by head of government*". Some connected entities are learned well for relation "*Opposite of*". We list 5 aligned sentences for the three relations, and report the Google Ngrams (year 2019)[8] of entities to show their popularity. We find that CF often happens to entities with large Google Ngrams such as "*left*" and "*President*". And CR often happens to entities with small Google Ngrams. Well-learned entities are also not very popular.

Can we improve the KI quality by simply increasing the size of our dataset? Above results analyze KI in LMs in different ways. However, there is a key question: can we simply improve the quality of KI by increasing the amount of our aligned training corpora? We try to answer this question in Figure 5 which plots correlation between the attention coefficients and the number of aligned sentences for knowledge triples in the dataset. We find that the *Pearson correlation* between the two is -0.0055. There is no positive correlation between the two variables. This implies there is no apparent positive relationship between the KI quality and the size of the KI dataset. It suggests that simply increasing the size of the aligned dataset alone may not improve KI, but, we might need more fundamental advances to push the state-of-the-art in KI forward.

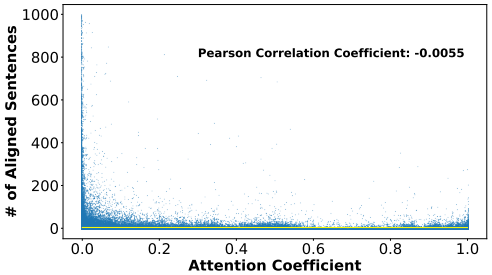


Figure 5: The correlation between the attention coefficient of the knowledge triple and its aligned sentence number. There is no correlation between them.

5 Limitations and Future Work

In this paper, we illustrate that the graph attention can be used to interpret the KI process in knowledge-enhanced LMs. Thus, we propose a Graph Convolutional Simulator (GCS) that is capable of interpreting both entity-wise and triple-wise knowledge-enhanced LMs. In our experiments, we verify GCS and use it to obtain some interesting findings about two popular knowledge-enhanced LMs. There are some limitations of our work. For example, we only consider relational knowledge in our work, and use it to justify our choice of GCS with one graph convolutional layer but based on the Theorem 1, this may not be sufficient to approximate the transformation for higher order knowledge. Finally, GCS only provides a way to interpret the knowledge integration. Once we have an understanding of the KI process, improving the integration quality remains an important challenge.

[8]https://books.google.com/ngrams

REPRODUCIBILITY STATEMENT

We will publish the code, as well as the interpretation results after the review process. The design of GCS can be found in §3. The implementation details about the KI for K-Adapter and ERNIE can be found in Appendix E. And details about GCS can be found in Appendix F.

ETHICS STATEMENT

While our probe models are not tuned for any specific real-world application, our methods could be used in sensitive contexts such as legal or health-care settings; and it is essential that any work that builds on our approaches undertake extensive quality-assurance and robustness testing before using it in their setting.

REFERENCES

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *NAACL-HLT*, pp. 3554–3565. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.278. URL https://doi.org/10.18653/v1/2021.naacl-main.278.

Arindam Banerjee. On bayesian bounds. In William W. Cohen and Andrew W. Moore (eds.), *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pp. 81–88. ACM, 2006. doi: 10.1145/1143844.1143855. URL https://doi.org/10.1145/1143844.1143855.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, R. Devon Hjelm, and Aaron C. Courville. Mutual information neural estimation. In Jennifer G. Dy and Andreas Krause (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 530–539. PMLR, 2018. URL http://proceedings.mlr.press/v80/belghazi18a.html.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Neural Information Processing Systems*, pp. 2787–2795, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2014. URL http://arxiv.org/abs/1312.6203.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15868–15876, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/71ee911dd06428a96c143a0b135041a4-Abstract.html.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. Ultra-fine entity typing. In Iryna Gurevych and Yusuke Miyao (eds.), *ACL*, pp. 87–96. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1009. URL https://aclanthology.org/P18-1009/.

Pedro Colon-Hernandez, Catherine Havasi, Jason B. Alonso, Matthew Huggins, and Cynthia Breazeal. Combining pre-trained language models and structured knowledge. *CoRR*, abs/2101.12294, 2021. URL https://arxiv.org/abs/2101.12294.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 5(4):455, 1992. doi: 10.1007/BF02134016. URL https://doi.org/10.1007/BF02134016.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Neural Information Processing Systems*, pp. 3837–3845, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *NAACL-HLT*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *CoRR*, abs/2106.15110, 2021. URL https://arxiv.org/abs/2106.15110.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Kôiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *LREC*. European Language Resources Association (ELRA), 2018. URL http://www.lrec-conf.org/proceedings/lrec2018/summaries/632.html.

Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An (eds.), *CIKM*, pp. 1625–1628. ACM, 2010. doi: 10.1145/1871437.1871689. URL https://doi.org/10.1145/1871437.1871689.

Guoji Fu, Yifan Hou, Jian Zhang, Kaili Ma, Barakeel Fanseu Kamhoua, and James Cheng. Understanding graph neural networks from graph signal denoising perspectives. *CoRR*, abs/2006.04386, 2020. URL https://arxiv.org/abs/2006.04386.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *NAACL-HLT*, pp. 4129–4138. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1419. URL https://doi.org/10.18653/v1/n19-1419.

Yifan Hou and Mrinmaya Sachan. Bird's eye: Probing for linguistic graph structures with a simple information-theoretic approach. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *ACL/IJCNLP*, pp. 1844–1859. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.145. URL https://doi.org/10.18653/v1/2021.acl-long.145.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *ACL*, pp. 3651–3657. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1356. URL https://doi.org/10.18653/v1/p19-1356.

Prakhar Kaushik, Alex Gain, Adam Kortylewski, and Alan L. Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *CoRR*, abs/2102.11343, 2021. URL https://arxiv.org/abs/2102.11343.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7090–7099, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/ea9268cb43f55d1d12380fb6ea5bf572-Abstract.html.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL http://arxiv.org/abs/1612.00796.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Xiao Ling, Sameer Singh, and Daniel S. Weld. Design challenges for entity linking. *Trans. Assoc. Comput. Linguistics*, 3:315–328, 2015. URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/528.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *NAACL-HLT*, pp. 1073–1094. Association for Computational Linguistics, 2019a. doi: 10.18653/v1/n19-1112. URL https://doi.org/10.18653/v1/n19-1112.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pp. 2901–2908. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5681.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b. URL http://arxiv.org/abs/1907.11692.

Yinquan Lu, Haonan Lu, Guirong Fu, and Qun Liu. KELM: knowledge enhanced pre-trained language representations with message passing on hierarchical relational graphs. *CoRR*, abs/2109.04223, 2021. URL https://arxiv.org/abs/2109.04223.

Ilsang Ohn and Yongdai Kim. Smooth function approximation by deep neural networks with general activation functions. *Entropy*, 21(7):627, 2019. doi: 10.3390/e21070627. URL https://doi.org/10.3390/e21070627.

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012. URL https://doi.org/10.1016/j.neunet.2019.01.012.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *EMNLP-IJCNLP*, pp. 43–54. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1005. URL https://doi.org/10.18653/v1/D19-1005.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *EMNLP-IJCNLP*, pp. 2463–2473. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1250. URL https://doi.org/10.18653/v1/D19-1250.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *ACL*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.420. URL https://doi.org/10.18653/v1/2020.acl-main.420.

Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Trans. Signal Process.*, 62(12):3042–3054, 2014. doi: 10.1109/TSP.2014.2321121. URL https://doi.org/10.1109/TSP.2014.2321121.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. *CoRR*, abs/2010.00577, 2020. URL https://arxiv.org/abs/2010.00577.

Kiran Koshy Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *CoRR*, abs/1803.03735, 2018. URL http://arxiv.org/abs/1803.03735.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Neural Information Processing Systems*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of ACL/IJCNLP*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 1405–1418. Association for Computational Linguistics, 2021a. doi: 10.18653/v1/2021.findings-acl.121. URL https://doi.org/10.18653/v1/2021.findings-acl.121.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194, 2021b. URL https://transacl.org/ojs/index.php/tacl/article/view/2447.

Liang Yao, Chengsheng Mao, and Yuan Luo. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193, 2019. URL http://arxiv.org/abs/1909.03193.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *ACL*, pp. 1441–1451. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1139. URL https://doi.org/10.18653/v1/p19-1139.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. Temporal reasoning on implicit events from distant supervision. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *NAACL-HLT*, pp. 1361–1371. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.107. URL https://doi.org/10.18653/v1/2021.naacl-main.107.

A NOTATIONS

Table 5: Notations and their descriptions

Notation	Description
$\mathcal{G}$	The knowledge graph for KI
$\mathcal{V}$	The set of entities/nodes of KG
$\mathcal{E}$	The set of relations/edges of KG
v_i	The entity/node indexed as i in the KG
t_i	The entity text attached on v_i
$\mathrm{LM}(\cdot)$	The language model, where the input is entity text, and the output is its representation
$\mathcal{N}_{v_i}$	The set of neighbors (entities/nodes) connected to v_i
$\mathcal{G}(v_i)$	The local graph structure in terms of v_i
$\mathbf{x}$	The random variable of the entity representation
$\boldsymbol{x}_i$	The entity representations of v_i
$\mathbf{g}$	The random variable of the local graph structure
$\mathrm{MI}(\cdot;\cdot)$	The mutual information between two random variables
$\boldsymbol{A}$	The adjacency matrix of KG
$\lvert\mathcal{V}\rvert$	The number of entities/nodes in KG
$\mathbb{R}$	The set of real numbers
$\boldsymbol{I}$	The identity matrix
$\boldsymbol{D}$	The degree matrix of KG
$\boldsymbol{L}_n$	The normalized Laplacian matrix
$\mathrm{diag}(\cdot)$	The diagonalization operation
$\boldsymbol{U}$	The matrix of eigenvectors
$\boldsymbol{\Lambda}$	The diagonal matrix of eigenvalues
λ_i	The i-th eigenvalue
$\boldsymbol{X}$	The set of entity representations in terms of $\mathcal{V}$
C	The dimension of entity representations; The number of channels
$\mathrm{GFT}(\cdot)$	The graph Fourier transformation
$\mathrm{RGFT}(\cdot)$	The inverse graph Fourier transformation
g_Θ	The graph filter parameterized by parameter Θ
$\boldsymbol{H}$	The entity representations given by a knowledge-enhanced LM
$\mathbf{h}$	The random variable of the entity representation given by a knowledge-enhanced LM
$f(\cdot)$	The mapping that can transform $\mathbf{x}$ to $\mathbf{h}$
ϵ	The error of the approximation
$\mathrm{sigmoid}(\boldsymbol{x})$	The Sigmoid function $\mathrm{sigmoid}(\cdot) = \frac{1}{1+e^{-\boldsymbol{x}}}$
n	The number of layers of the neural network for apporximation
$\boldsymbol{W}$	The weight matrix
$\boldsymbol{x}$	The input vector
$\boldsymbol{b}$	The bias
λ_0'	The minimum eigenvalue of the weight matrix $\boldsymbol{W}$
$\mathrm{MLP}_n(\cdot)$	The bijective MLP function
$\mathrm{GC}(\cdot,\cdot)$	The graph convolution function
GCS_{θ_1}	The GCS model parameterized by θ_1
$\mathcal{L}$	The objective of the optimization
$\boldsymbol{Z}$	The output of GCS, i.e., set of output entity representations
$\mathbf{z}$	The random variable of the output of GCS
$\sup$	The supremum value
T	A class of functions
$\mathcal{F}$	Any class of functions
Ω	The domain of a function
T_{θ_2}	A class of functions parameterized by θ_2, i.e., neural networks
$\mathbb{P}$	The probability distribution
$\mathbb{P}^{\lvert\mathcal{V}\rvert}$	The empirical distribution with $\lvert\mathcal{V}\rvert$ samples
$\mathrm{NN}_\sigma(\cdot\lvert\theta')$	The neural network with activation function $\sigma(\cdot)$ and parameterized by θ'
$\lvert\boldsymbol{U}\rvert$	The norm of matrix $\boldsymbol{U}$
$\boldsymbol{A}_n$	The normalized adjacency matrix
$\hat{\boldsymbol{X}}$	The ground-truth entity representations/node features
$\hat{\boldsymbol{X}}^*$	The variable matrix
$\mathbf{Tr}(\cdot)$	The trace of a matrix
ϵ_1, ϵ_2	The error bound of entity representations/node features and adjacency matrix
γ	The Lagrangian multiplier
$p(t)$	The characteristic polynomial for weight matrix $\boldsymbol{W}$
$\det(\cdot)$	The determinant of a matrix

B PROOF OF THEOREM 1

Proof. As aforementioned, the graph Fourier transformation $\mathrm{GFT}(\cdot)$ and its inverse transformation $\mathrm{RGFT}(\cdot)$ in terms of the KG $\mathcal{G}$ can be written as

$$\mathrm{GFT}(\boldsymbol{X}) = \boldsymbol{U}^T \boldsymbol{X}$$
$$\mathrm{RGFT}(\mathrm{GFT}(\boldsymbol{X})) = \boldsymbol{U}\mathrm{GFT}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{X} = \boldsymbol{X}.$$

The second equation can be derived since $\boldsymbol{U}$ is the set of eigenvectors of the normalized Laplacian matrix in terms of $\mathcal{G}$, which is orthogonal.

According to the universal approximation theorem (Cybenko, 1992), in general, we can use one-layer neural networks (arbitrary width) with the sigmoid activation function to fit any functions. Ohn & Kim (2019) bound the approximation with both the width and depth, and supports more activation functions. Based on the conclusion of Ohn & Kim (2019), we know that given a mapping $f'(\cdot)$, for any $\epsilon' > 0$, there exists a neural network parameterized by θ' s.t.

$$|f'(\cdot) - \mathrm{NN}_\sigma(\cdot|\theta')| < \epsilon'.$$

Note that there are some constraints about the input and the model architecture, i.e., width and depth. We leave out those details for simplicity. More details can be found in Ohn & Kim (2019).

Since $\mathbf{h}$ is obtained by integrating $\mathbf{g}$ into $\mathbf{x}$, we can simplify the mapping in the graph space by researching on the transformation from $\mathrm{GFT}(\mathbf{x})$ to $\mathrm{GFT}(\mathbf{h})$[9]. Assume the mapping satisfies $f'(\mathrm{GFT}(\mathbf{x})) = \mathrm{GFT}(\mathbf{h})$. Then we have

$$|f'(\mathrm{GFT}(\mathbf{x})) - \mathrm{NN}_\sigma(\mathrm{GFT}(\mathbf{x})|\theta')| < \epsilon'.$$

Consider that we have $f(\mathbf{x}, \mathbf{g}) = \mathbf{h} = \mathrm{RGFT}(f'(\mathrm{GFT}(\mathbf{x})))$. If we assign $\epsilon' = \frac{\epsilon}{|\boldsymbol{U}|} > 0$, we have

$$|\boldsymbol{U}| \cdot |f'(\mathrm{GFT}(\mathbf{x})) - \mathrm{NN}_\sigma(\mathrm{GFT}(\mathbf{x})|\theta')| < \epsilon.$$

Since we know that

$$\boldsymbol{U} \cdot f'(\mathrm{GFT}(\mathbf{x})) = \mathrm{RGFT}(f'(\mathrm{GFT}(\mathbf{x}))) = \mathbf{h} = f(\mathbf{x}, \mathbf{g}),$$

we have

$$|f(\mathbf{x}, \mathbf{g}) - \mathrm{RGFT}(\mathrm{NN}(\mathrm{GFT}(\mathbf{x})))| < |\boldsymbol{U}| \cdot |f'(\mathrm{GFT}(\mathbf{x})) - \mathrm{NN}_\sigma(\mathrm{GFT}(\mathbf{x})|\theta')| < \epsilon,$$

where $\mathrm{NN}(\cdot)$ is parameterized by θ' with activation function σ as $\mathrm{NN}_\sigma(\cdot|\theta')$. And without loss of generality, we assume it is composed of n layers. □

C GRAPH ATTENTION FOR KI INTERPRETATION (PROOF)

Proof. The basic idea of this proof can be found in Figure 3. We first simply prove that the graph Fourier transformation is bijective. Similarly, the nonlinear activation function can be proved bijective. Then, we show that information gain and loss can only happen in the linear function in graph space. After that, we briefly illustrate that linear function in graph space is graph convolution operation. Finally, we prove that graph attention works as edge denoising, and we can use it to interpret the KI.

$\mathrm{GFT}(\cdot)$**,** $\mathrm{RGFT}(\cdot)$**, and** $\mathrm{sigmoid}(\cdot)$ **are bijective.** Given two entity representations $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and the matrix of eigenvectors of the KG as $\boldsymbol{U}$, suppose that $\mathrm{GFT}(\boldsymbol{x}_i) = \mathrm{GFT}(\boldsymbol{x}_j)$. Then, we have

$$\boldsymbol{U}^T\boldsymbol{x}_i = \boldsymbol{U}^T\boldsymbol{x}_j.$$

[9] In next proof, we illustrate that the linear transformation in the graph space is graph convolution, which integrates the graph information into entities. Thus, in the graph space, we do not need to regard $\mathbf{g}$ as an input. More formally description can be found in Chen et al. (2019); Keriven & Peyré (2019).

Since $\boldsymbol{U}^T$ are set of eigenvectors and are by definition nonzero, we have

$$\boldsymbol{x}_i = \boldsymbol{x}_j.$$

If $\boldsymbol{x}_i = \boldsymbol{x}_j$, it is easy to get $\mathrm{GFT}(\boldsymbol{x}_i) = \mathrm{GFT}(\boldsymbol{x}_j)$. Thus, graph Fourier transformation is bijective.

As for the nonlinear activation function, since we consider neural networks composed of MLP layers, the activation function is $\mathrm{sigmoid}(\cdot)$ function. It is easy to find that its inverse function is $f(\boldsymbol{y}) = \ln(1 - \frac{1}{\boldsymbol{y}})$. Similarly, we can prove that it is bijective as well.

Information gain and loss can only happen in the linear function in graph space. Based on the invariance of MI (Kraskov et al., 2004), we have

$$\begin{aligned}
\mathrm{MI}(\mathbf{x}, \mathbf{g}) &= \mathrm{MI}(\mathrm{GFT}(\mathbf{x}), \mathbf{g}), \\
\mathrm{MI}(\mathbf{x}, \mathbf{g}) &= \mathrm{MI}(\mathrm{RGFT}(\mathbf{x}), \mathbf{g}), \\
\mathrm{MI}(\mathbf{x}, \mathbf{g}) &= \mathrm{MI}(\mathrm{sigmoid}(\mathbf{x}), \mathbf{g}).
\end{aligned}$$

Since we know that

$$\mathrm{MI}(\mathbf{h}, \mathbf{g}) - \mathrm{MI}(\mathbf{x}, \mathbf{g}) > 0,$$

and the neural network can well approximate the mapping, we have

$$\begin{aligned}
\mathrm{MI}(\mathbf{h}, \mathbf{g}) - \mathrm{MI}(\mathbf{x}, \mathbf{g}) &\approx \mathrm{MI}(\mathrm{RGFT}(\mathrm{NN}(\mathrm{GFT}(\mathbf{x}))), \mathbf{g}) - \mathrm{MI}(\mathbf{x}, \mathbf{g}) \\
&= \mathrm{MI}(\mathrm{NN}(\mathrm{GFT}(\mathbf{x})), \mathbf{g}) - \mathrm{MI}(\mathrm{GFT}(\mathbf{x}), \mathbf{g}) > 0.
\end{aligned}$$

If we write $\mathrm{NN}(\cdot)$ with n MLP layers as $n \times \sigma(\mathrm{Linear}(\cdot))$, we have

$$\mathrm{MI}(n \times \sigma(\mathrm{Linear}(\mathrm{GFT}(\mathbf{x})), \mathbf{g}) - \mathrm{MI}(\mathrm{GFT}(\mathbf{x}), \mathbf{g}) > 0.$$

Recursively, it is easy to get that MI only changes in the $\mathrm{Linear}(\cdot)$ functions.

The linear function in the graph space is the graph convolution operation. Even if many existing works (Sandryhaila & Moura, 2014; Bruna et al., 2014; Kipf & Welling, 2017) have provided clear descriptions, we simply re-illustrate it under the multi-channel setting. Consider the graph filter in Bruna et al. (2014) as an exmaple.

For a linear function $f(\boldsymbol{x}) = \boldsymbol{W} \times \boldsymbol{x}$, its weight matrix $\boldsymbol{W} \in \mathbb{R}^{F \times C}$ is parameterized by $\Theta \in \mathbb{R}^{F \times C}$. If the parameters are not shared for all nodes, the input $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times C}$ can be rescaled in $\mathbb{R}^{|\mathcal{V}| \times C \times 1}$, and the weight matrix is $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times F \times C}$ parameterized by $\Theta \in \mathbb{R}^{F \times C \times |\mathcal{V}|}$. The output of this linear function is mapped in $\mathbb{R}^{|\mathcal{V}| \times F}$.

Consider the signal in graph convolution, i.e., all $\boldsymbol{x}$ in $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times C}$. Since parameters are not shared (Bruna et al., 2014), for one graph filter, the parameters in g_Θ is in $\mathbb{R}^{C \times |\mathcal{V}| \times |\mathcal{V}|}$ that is parameterized by $\Theta \in \mathbb{R}^{C \times |\mathcal{V}|}$ with simple diagonalization. If we have F different graph filters for the convolution, g_Θ is in $\mathbb{R}^{F \times C \times |\mathcal{V}| \times |\mathcal{V}|}$ that is parameterized by $\Theta \in \mathbb{R}^{F \times C \times |\mathcal{V}|}$. Here, the graph Fourier transformation of $\boldsymbol{X}$ is $\mathrm{GFT}(\boldsymbol{X}) \in \mathbb{R}^{|\mathcal{V}| \times C}$, which can be rescaled in $\mathbb{R}^{1 \times |\mathcal{V}| \times C \times 1}$ with simple diagonalization. The output is in $\mathbb{R}^{F \times |\mathcal{V}| \times |\mathcal{V}| \times 1}$. Note that since the parameters in the graph filter is diagonalized, we can rescale the output in $\mathbb{R}^{|\mathcal{V}| \times F}$.

If we regard the weight matrix $\boldsymbol{W}$ as the parameters in the graph filter g_Θ, the input matrix $\boldsymbol{X}$ as the signal, obviously, the linear function in the graph space is the graph convolution operation.

Graph attention works as edge denoising[10]. Graph attention works as a better graph convolution filter, since it can adaptively learn the optimal convolution weights (i.e., attention coefficients). Consider a graph signal denoising problem that we aim to extract the ground-truth node features $\hat{\boldsymbol{X}}$ and edge weights $\hat{\boldsymbol{A}}_n$ from a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A}_n)$ with noise in both node features $\boldsymbol{X}$ and edge weights $\boldsymbol{A}_n$. Here, $\boldsymbol{A}_n$ is the normalized adjacency matrix $\boldsymbol{A}_n = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. To this end, we formulate the optimization problem under the assumption that the ground-truth node features $\hat{\boldsymbol{X}}$ are smooth w.r.t the ground-truth adjacency matrix $\hat{\boldsymbol{A}}_n$ and the noise in the graph can be upper-bounded:

$$\begin{aligned}
\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{A}}_n^* = \underset{\hat{\boldsymbol{X}}, \hat{\boldsymbol{A}}_n}{\mathrm{argmin}} & \mathrm{Tr}\left(\hat{\boldsymbol{X}} \hat{\boldsymbol{L}}_n^T \hat{\boldsymbol{X}}\right) \\
\text{s.t. } & \|\hat{\boldsymbol{X}} - \boldsymbol{X}\|_2^2 \leq \epsilon_1, \\
& \|\hat{\boldsymbol{A}}_n - \boldsymbol{A}_n\|_2^2 \leq \epsilon_2,
\end{aligned} \tag{7}$$

[10]The detailed proof can be found in Fu et al. (2020).

where $\hat{\boldsymbol{L}} = \boldsymbol{I} - \hat{\boldsymbol{A}}$, $\epsilon_1, \epsilon_2 \in \mathbb{R}$, are the level of noise in node features and edge weights, respectively. $\mathrm{Tr}(\cdot)$ indicates the trace of a matrix. By Lagrange multipliers methods, we can obtain the solution as following:

$$\hat{\boldsymbol{X}}^* = \frac{\gamma}{1+\gamma}\left(\boldsymbol{I} - \frac{1}{1+\gamma}\hat{\boldsymbol{A}}_n^*\right), \tag{8}$$

$$\hat{\boldsymbol{A}}_n^* = \boldsymbol{A}_n + \sqrt{\epsilon_2}\frac{\hat{\boldsymbol{X}}^*\hat{\boldsymbol{X}}^{*\top}}{\|\hat{\boldsymbol{X}}\|_2^2}, \tag{9}$$

where $\gamma > 0$ is the Lagrangian multiplier. Note that the attention coefficients of GAT (Velickovic et al., 2018) and AGNN (Thekumparampil et al., 2018) are obtained by (without less of generality, we show the results in the first-layer) equation 10 and equation 11, respectively:

$$a_{i,j} = \mathrm{softmax}\left(\mathrm{leakyReLU}\left(\mathbf{a}^\top\left[\boldsymbol{W}\boldsymbol{X}_i \| \boldsymbol{W}\boldsymbol{X}_j\right]\right)_{j\in\mathcal{N}_i\cup\{i\}}\right), \tag{10}$$

$$a_{i,j} = \mathrm{softmax}\left(\left[\beta\frac{\boldsymbol{H}_i^\top\boldsymbol{H}_j}{\|\boldsymbol{H}_i\|\|\boldsymbol{H}_j\|}\right]_{j\in\mathcal{N}_i\cup\{i\}}\right), \tag{11}$$

where $\boldsymbol{H} = \mathrm{ReLU}(\boldsymbol{X}\boldsymbol{W})$, $\mathbf{a}$, $\boldsymbol{W}$ in equation 10, and β, $\boldsymbol{W}$ in equation 11 are learnable parameters. The attention coefficents of GAT and AGNN are then used as the weights of aggregating the neighbohood information of nodes. As we can see that equation 9, equation 10, and equation 11 are in a form of measuring the similarity between paired node features. Similar to the denoised edge weights obtained in equation 9, the attention coefficents (i.e. the aggregation weights) between a node and its neighborhoods are proportional to the similarity of their node embeddings. Therefore, the attention coefficients of GAT and AGNN can be regarded as the results of denoised weights on the existing edges in a graph, i.e., the graph attentions are implicitly denoising the edge weights.

Therefore, in our case, we can use attention coefficients in graph attention in graph convolution layer to interpret the KI process. As for the forgetting situation, equally, we can use the attention coefficients on the self-loop edges for interpretation.

□

D MLP CAN BE BIJECTIVE (PROOF)

Theorem 2. *Give an MLP layer denoted as* $\mathrm{MLP}(\boldsymbol{x}) = \mathrm{sigmoid}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. *If* $\boldsymbol{W}$ *is a square matrix, there exist a constant* $\lambda_0' > 0$ *that for any* $0 < \epsilon < \lambda_0'$, *the function below is bijective:*

$$\mathrm{MLP}_n(\boldsymbol{x}) = \mathrm{sigmoid}((\boldsymbol{W} - \epsilon\boldsymbol{I})\boldsymbol{x} + \boldsymbol{b}). \tag{12}$$

Proof. We first prove that two bijective function compositions are still bijective. Then, we prove that adding a small noise on MLP weight matrix can make it bijective.

Give two function $f_1(\cdot)$ and $f_2(\cdot)$. Suppose they are injective and suppose $f_1(f_2(\boldsymbol{x})) = f_1(f_2(\boldsymbol{y}))$. Since we know that $f_1(\cdot)$ is injective, we have $f_2(\boldsymbol{x}) = f_2(\boldsymbol{y})$. Similarly, since $f_2(\cdot)$ is injective, we have $\boldsymbol{x} = \boldsymbol{y}$. Thus $f_1(f_2(\cdot))$ is injective. Suppose $f_1(\cdot)$ and $f_2(\cdot)$ are surjective and $\boldsymbol{z} \in C$. Since we know that $f_1(\cdot)$ is surjective, there exists a set of $\boldsymbol{y} \in B$ with $f_1(\boldsymbol{y}) = \boldsymbol{z}$. Similarly, since $f_2(\cdot)$ is surjective, there exists a set of $\boldsymbol{x} \in A$ with $f_2(\boldsymbol{x}) = \boldsymbol{y}$. Then, we have $\boldsymbol{z} = f_1(f_2(\boldsymbol{x}))$ and so $\boldsymbol{z}$ is onto $f_1(f_2(\cdot))$. Thus, $f_1(f_2(\cdot))$ is surjective. Therefore, if $f_1(\cdot)$ and $f_2(\cdot)$ are bijective, $f_1(f_2(\cdot))$ is also bijective.

To prove that the special MLP is bijective, consider an MLP function as

$$\mathrm{MLP}(\boldsymbol{x}) = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}),$$

where $\boldsymbol{W} \in \mathbb{R}^{C\times C}$ is the weight matrix and $\boldsymbol{b} \in \mathbb{R}^C$ is the bias. Let

$$p(t) = \prod_{i=1}^{C}(\lambda_i' - t)$$

be the characteristic polynomial for weight matrix $\boldsymbol{W}$. Here λ_i' are eigenvalues of matrix $\boldsymbol{W}$. Without loss of generality, let $|\lambda_0'| = \min_i |\lambda_i'|$. Then, we know that for any constant $0 < \epsilon < |\lambda_0'|$, we have

$$\det(\boldsymbol{W} - \epsilon \boldsymbol{I}) = p(\epsilon) \neq 0.$$

Thus, if the perturbation ϵ is small enough, the perturbed matrix $\boldsymbol{W}' = \boldsymbol{W} - \epsilon \boldsymbol{I}$ is nonsingular. Consider the fact that the nonlinear activation function $\sigma(\cdot)$ is $\mathrm{sigmoid}(\cdot)$ function, which is bijective. Therefore, the special MLP function $\mathrm{MLP}_n(\cdot)$ is bijective. And there is no information loss. □

E DETAILS OF LMS

To ensure that the experiment settings are fair, we set hyperparameters as the default values. For K-Adapter, the code and hyperparameters for KI that we use are from the official projects[11] published by the authors (Wang et al., 2021a). The only two differences are that: we use PyTorch *float 32* instead of *float 16* due to some bugs[12], and we use 4 NVIDIA Tesla V100 GPUs for integration. As for the downstream tasks, all the hyperparameters are consistent with the official project: either they are given in the project or in the README. In the same way, *float 32* and 4 NVIDIA Tesla V100 GPUs are chosen to make sure that the comparison is fair.

For ERNIE, things are the same. All hyperparameters for KI and downstream tasks are set as their default values[13]. Similarly, *float 16* of PyTorch is changed to *float 32*, and we do the integration and downstream task with 4 NVIDIA Tesla V100 GPUs. Note that the dataset that ERNIE used for KI is wikipedia, since the code is to fetch latest version of it, the data that we use could be slightly different. Therefore, for both ERNIE and K-Adapter, to ensure the fairness, we reproduce their KI, and report the results of reproduced models instead of results provided in their papers.

Table 6: Statistics of T-REx-rc and Wikidata

Datasets \ Statistics	# of entities	# of triples	# of aligned sentences	# of entities (optimization)	# of triples (optimization)
T-REx-rc	781,275	1,282,504	5,565,478	-	-
Wikidata	3,275,534	12,849,311	-	1,344,393	3,240,272

The datasets that K-Adapter and ERNIE use are T-REx-rc and Wikidata, some statistics of them are given in Table 6.

F DETAILS OF GCS

In this section, we first introduce details of implementing GCS in practice. Then, we illustrate the way to use GCS to interpret KI process.

In practice, GCS is composed of 5 layers. The first three layers are: bijective MLP layer, graph convolutional layer, and another bijective MLP layer. The first two MLP layers are designed to simulate graph Fourier transformation. The last two layers are MLPs for MI maximization. As for bijective MLP layers, since weight matrices in them are square matrices, the dimension would remain unchanged: 1024 for K-Adapter and 768 for ERNIE. The last two MLP layers may not be bijective, where the dimension would be first reduced to 64, then to 1 for MI maximization. The nonlinear activation functions are all set as $\mathrm{ELU}(\cdot)$ function, which is also bijective. The learning rate is set as $1e^{-3}$, and the dropout rate of the first two MLP layers is 0.2.

As for the graph attention, to make sure the results are stable, we apply multi-head attention mechanism, where the number of attention head is set as 8. Entity representations are first embedded into a space with the dimension as 64. Then, the embedded representations are used to calculate the attention coefficients. Note that since the purpose is to interpret and analyze the KI process, we do not split datasets for KI. Considering that our objective is to maximize MI, and GCS model is very

[11] https://github.com/microsoft/K-Adapter

[12] Due to the old cuda version, we did not successfully install the apex library provided by NVIDIA, thus, the *float 16* of PyTorch cannot be implemented.

[13] https://github.com/thunlp/ERNIE

simple for the large KGs, overfitting is unlikely to happen. Thus, we optimize GCS for the whole datasets. Specifically, for K-Adapter, the whole KG is used for optimization, and the results are used for interpretation. And for ERNIE, since the KG is very large, we sample a small subgraph with $1,344,393$ entities and $3,240,272$ triples for optimization, and implement the optimized GCS on the whole KG for interpretation as shown in Table 6.

For interpretation, we mainly use the attention coefficients on edges and self-loops to analyze the KI in terms of triples and entities. Different from Schlichtkrull et al. (2020) that specially designs a discrete function to mask edges that are not important, we simply introduce a temperature hyperparameter t and set it as $t = 0.1$ to make the attention coefficient distribution hard[14]. Thus, knowledge can be well clustered into learned and unlearned.

Attention coefficients on edges indicate the "information flow" on the KG to transform entity representations from pretrained LM to knowledge-enhanced LM. Large values on edges means that the corresponding triples play an important role in the transformation. Thus, we regard these knowledge triples as learned ones. Based on the empirical distributions in Figure 4, we set the threshold as 0.1 to separate triples as learned and unlearned ones. Note that users may choose a different threshold in terms of the temperature t and the empirical distribution. Attention coefficients on self-loops indicate the "information change" of entities. Large values represent that the change is significant, which implies that much new knowledge is successfully integrated into entity representations. Similarly, we set thresholds as $0.1, 0.4, 0.6, 0.9$. When $a_{i,i} < 0.1$, it means that nearly all original information is dropped during the KI, which is called catastrophic forgetting (CF). When $a_{i,i} > 0.9$, we can interpret that nearly all original information is retained, which is called catastrophic remembering (CR). Based on the empirical distributions, we simply regard entities with $0.4 < a_{i,i} < 0.6$ are well-integrated ones.

G Additional Statistics of Table 3

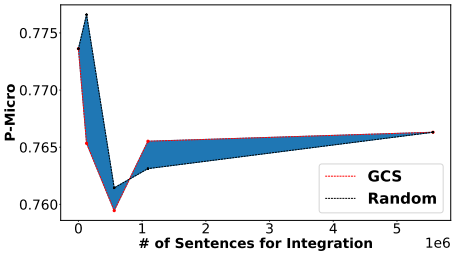
Table 7: The number of aligned sentences for relations.

Statistics / Relation label	# of triples
Place of birth	134,976
Part of	134,999
Date of death	135,190
Date of birth	135,169
Located in the administrative territorial entity	135,055
Country	135,147
Total	5,565,478

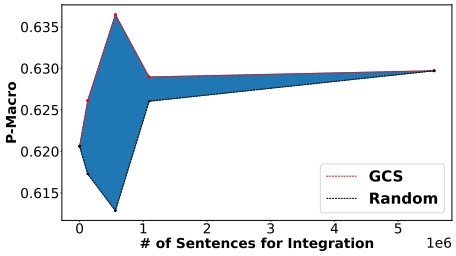
H Detailed Results for GCS verification

We gradually drop out sentences aligned to knowledge triples whose attention coefficients are small for KI. For comparison, we randomly drop same number of sentences named the random strategy. We can see that dropping out KI datasets based on the interpretation results of GCS significantly outperforms the random strategy. It further supports that GCS can effectively interpret the KI situation for the K-Adapter.

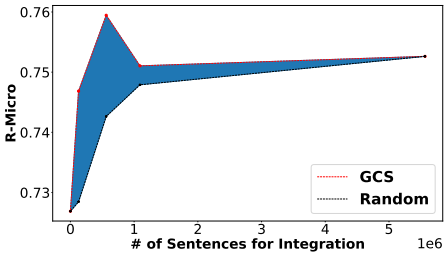
[14]Note that the principle of hyperparameter selection is to maximize the MI, i.e., objective function. Users may select appropriate hyperparameters depending on the situation.



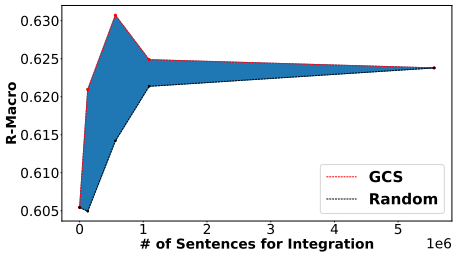
(a) Precision-Micro score



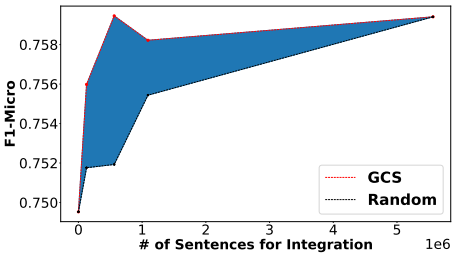
(b) Precision-Macro score



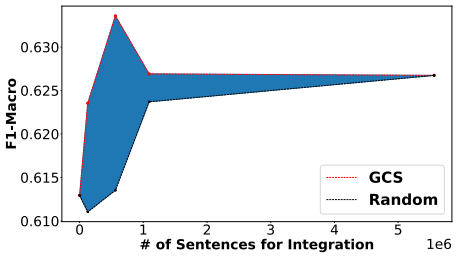
(c) Recall-Micro score



(d) Recall-Macro score



(e) F1-Micro score



(f) F1-Macro score

Figure 6: Detailed results of K-Adapter on the OpenEntity dataset, where we use different number of sentences to integrate knowledge into K-Adapter. For GCS, we integrate knowledge for K-Adapter using sentences aligned with triples whose attention coefficients are larger than $\{0.0,\ 0.01,\ 0.1,\ 0.9,\ 1.0\}$, where the corresponding number of sentences are $\{5,565,478,\ 1,091,152,\ 561,687,\ 127,728,\ 0\}$. Same number of sentences are chosen randomly for comparison as the random strategy.