KALA: Knowledge-Augmented Language Model Adaptation

Anonymous ACL submission

Abstract

Pre-trained language models (PLMs) have achieved remarkable success on various natural language understanding tasks. Simple fine-tuning of PLMs, on the other hand, might be suboptimal for domain-specific tasks because they cannot possibly cover knowledge from all domains. While adaptive pre-training of PLMs can help them obtain domain-specific knowledge, it requires a large training cost. Moreover, adaptive pre-training can harm the PLM's performance on the downstream task by causing catastrophic forgetting of its general knowledge. To overcome such limitations of adaptive pre-training for PLM adaption, we propose a novel domain adaption framework for PLMs coined as **K**nowledge-**A**ugmented **L**anguage model **A**daptation (**KALA**), which modulates the intermediate hidden representations of PLMs with domain knowledge, consisting of entities and their relational facts. We validate the performance of our KALA on question answering and named entity recognition tasks on multiple datasets across various domains. The results show that, despite being computationally efficient, our KALA largely outperforms adaptive pre-training.

1 Introduction

Pre-trained Language Models (PLMs) (Devlin et al., 2019; Brown et al., 2020) have shown to be effective on various Natural Language Understanding (NLU) tasks. Although PLMs aim to address diverse downstream tasks from various data sources, there have been considerable efforts to adapt the PLMs to specific *domains* —distributions over the language characterizing a given topic or genre (Gururangan et al., 2020)— for which the acquisition of domain knowledge is required to accurately solve the downstream tasks (e.g., Biomedical Named Entity Recognition (Dogan et al., 2014)).

This problem, known as *Language Model Adaptation*, can be viewed as a transfer learning problem (Yosinski et al., 2014; Ruder, 2019) under domain shift, where the model is pre-trained on the general domain and the labeled distribution is available for the target domain-specific task. The most prevalent approach to this problem is adaptive pre-training (Figure 2a) which further updates all parameters of the PLM on a large domain-specific or curated task-specific corpus, with the same pre-training strategy (e.g., masked language modeling) before fine-tuning it on the downstream task (Beltagy et al., 2019; Lee et al., 2020; Gururangan et al., 2020). This continual pre-training of a PLM on the target domain corpus allows it to learn the distribution of the target domain, resulting in improved performance on domain-specific tasks (Howard and Ruder, 2018; Han and Eisenstein, 2019).

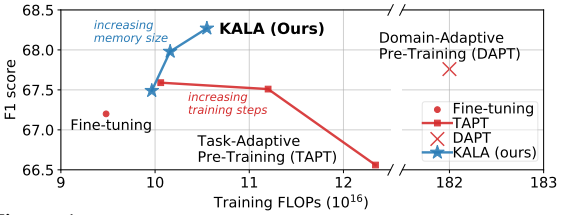


Figure 1: F1 Score and Training FLOPs for different methods on Question Answering (NewsQA). Note that DAPT uses about 112 times larger data for adaptation. Details are in §5.2

While it has shown to be effective, adaptive pre-training has obvious drawbacks. First, it is computationally inefficient. Although a PLM becomes more powerful with the increasing amount of pre-training data (Gururangan et al., 2020), further pre-training on the additional data requires larger memory and computational cost as the dataset size grows (Bai et al., 2021). Besides, it is difficult to adapt the PLM to a new domain without forgetting the general knowledge it obtained from the initial pretraining step, since all pre-trained parameters are continually updated to fit the domain-specific corpus during adaptive pre-training (Chen et al., 2020). This *catastrophic forgetting* of the task-general knowledge may lead to the performance degradation on the downstream tasks. In Figure 1, we show that adaptive pre-training with more training steps could lead to performance degeneration.

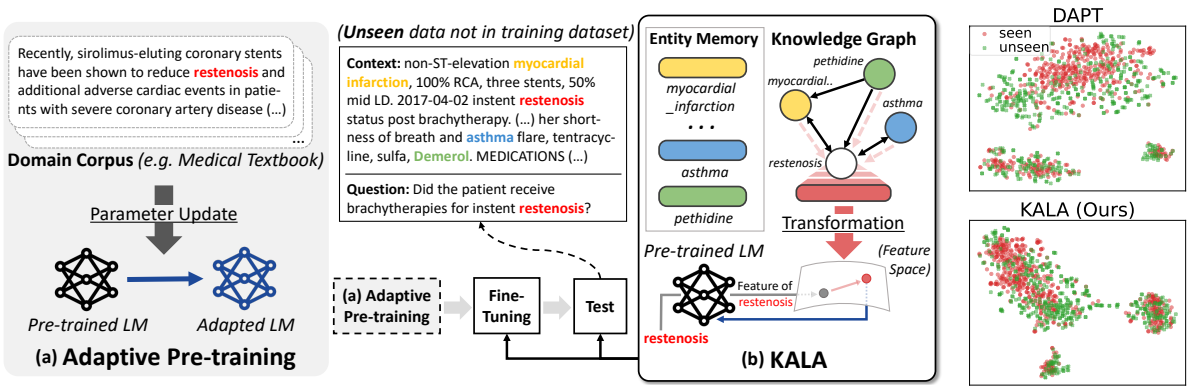


Figure 2: **Concepts (Left).** (a) Adaptive Pre-training updates whole parameters of the PLM through further pre-training on the domain corpus. (b) Our method KALA integrates the external knowledge so that the PLM adapts to the target domain **only with fine-tuning**, which is realized by the affine transformation on the intermediate feature. **Visualization of the contextualized representation from the PLM for seen and unseen entities (Right).** Our KALA framework embeds the unseen entities on the embedding space of seen entities by representing them with their relational knowledge over the graph, while the strong DAPT baseline (Gururangan et al., 2020) cannot appropriately handle unseen entities that are not given for task fine-tuning.

Thus, it would be preferable if we could adapt the PLM to the domain-specific task without costly adaptive pre-training. To this end, we aim to integrate the domain-specific knowledge into the PLM directly during the task-specific fine-tuning step, as shown in Figure 2b, eliminating the adaptive pre-training stage. Specifically, we first note that **entities** and **relations** are core building blocks of the domain-specific **knowledge** that are required to solve for the domain-specific downstream tasks. Clinical domain experts, for example, are familiar with medical terminologies and their complex relations. Then, to represent the domain knowledge consisting of entities and relations, we introduce the *Entity Memory*, which is the source of entity embeddings but independent of the PLM parameters (See Entity Memory in Figure 2b). Then, we further exploit the relational structures of the entities by utilizing a Knowledge Graph (KG), which denotes the factual relationships between entities, as shown in Knowledge Graph of Figure 2b.

The remaining step is how to integrate the knowledge into the PLM during fine-tuning. To this end, we propose a novel layer named Knowledge-conditioned Feature Modulation (KFM, §3.2), which scales and shifts the intermediate hidden representations of PLMs by conditioning them with retrieved knowledge representations. This knowledge integration scheme has several advantages. First, it does not modify the original PLM architecture, and thus could be integrated into any PLMs regardless of their architectures. Also, it only requires marginal computational and memory overhead, while eliminating the need of excessive further pre-training (Figure 1). Finally, it can effectively handle unseen entities with relational knowledge from the KG, which are suboptimally embedded by adaptive pre-training. For example, as shown in Figure 2, an entity *restenosis* does not appear in the training dataset for fine-tuning, thus adaptive pre-training only implicitly infers them within the context from the broad domain corpus. However, we can explicitly represent the unknown entity by aggregating the representations of known entities in the entity memory (i.e., in Figure 2, neighboring entities, such as *asthma* and *pethidine*, are used to represent the unseen entity *restenosis*).

We combine all the previously described components into a novel language model adaptation framework, coined as **K**nowledge-**A**ugmented **L**anguage model **A**daptation (KALA) (Figure 3). We empirically verify that **KALA** improves the performance of the PLM over adaptive pre-training on various domains with two knowledge-intensive tasks: Question Answering (QA) and Named Entity Recognition (NER). Our contribution is threefold:

- We propose a novel LM adaptation framework, which augments PLMs with entities and their relations from the target domain, during fine-tuning without any further pre-training. To our knowledge, this is the first work that utilizes the structured knowledge for language model adaptation.
- To reflect structural knowledge into the PLM, we introduce a novel layer which scales and shifts the intermediate PLM representations with the entity representations contextualized by their related entities according to the KG.
- We show that our KALA significantly enhances the model's performance on domain-specific tasks while being significantly more efficient over existing LM adaptation methods.

2 Related Work

Language Model Adaptation Nowadays, transfer learning (Howard and Ruder, 2018) is a dominant approach for solving Natural Language Understanding (NLU) tasks. This strategy first pre-trains a Language Model (LM) on a large and unlabeled corpus, then fine-tunes it on downstream tasks with labeled data (Devlin et al., 2019). While this scheme alone achieves impressive performance on various NLU tasks, adaptive pre-training of the PLM on a domain-specific corpus helps the PLM achieve better performance on the domain-specific tasks. For example, Lee et al. (2020) demonstrated that a further pre-trained LM on biomedical documents outperforms the original LM on biomedical NLU tasks. Also, Gururangan et al. (2020) showed that adaptive pre-training of the PLM on the corpus of a target domain (Domain-adaptive Pre-training; DAPT) or a target task (Task-adaptive Pre-training; TAPT) improves its performance on domain-specific tasks.

Knowledge-aware LM Accompanied with increasing sources of knowledge (Vrandecic and Krötzsch, 2014), some prior works have proposed to integrate external knowledge into PLMs, to enhance their performance on tasks that require structured knowledge. For instance, ERNIE (Zhang et al., 2019) and KnowBERT (Peters et al., 2019) incorporate entities as additional inputs in the pre-training stage to obtain a knowledge-aware LM, wherein a pre-trained knowledge graph embedding from Wikidata (Vrandecic and Krötzsch, 2014) is used to represent entities. Entity-as-Experts (Févry et al., 2020) and LUKE (Yamada et al., 2020) use the entity memory that is pre-trained along with the LMs from scratch. ERICA (Qin et al., 2021) further uses the fact consisting of entities and their relations in the pre-training stage of LMs from scratch. Previous works aim to integrate external knowledge into the LMs during the pre-training step to obtain a universal knowledge-aware LM that requires additional parameters for millions of entities. In contrast to this, our framework aims to efficiently modify a general PLM for the domain-specific task with a linear modulation layer scheme discussed in Section 3.2, during fine-tuning.

3 Method

3.1 Problem Statement

Our goal is to solve Natural Language Understanding (NLU) tasks for a specific domain, with a knowledge-augmented Language Model (LM). We first introduce the NLU tasks we target, followed by the descriptions of the proposed knowledge-augmented LM. After that, we formally define the ingredients for structured knowledge integration.

NLU tasks The goal of an NLU task is to predict the label $\boldsymbol{y}$ of the given input instance $\boldsymbol{x}$, where the input $\boldsymbol{x}$ contains the sequence of tokens (Devlin et al., 2019): $\boldsymbol{x} = [w_1, w_2, \ldots, w_{|\boldsymbol{x}|}]$. Then, given a training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$, the objective is to maximize the log-likelihood as follows:

$$\max_{\theta} \mathcal{L}(\theta) := \max_{\theta} \sum_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x};\theta),$$

$$p(\boldsymbol{y}|\boldsymbol{x};\theta) = g(\boldsymbol{H};\theta_g),\ \boldsymbol{H} = f(\boldsymbol{x};\theta_f),$$

where f is an encoder of the PLM which outputs contextualized representation $\boldsymbol{H}$ from $\boldsymbol{x}$, and g is a decoder which models the probability distribution p of the label $\boldsymbol{y}$, with trainable parameters $\theta = (\theta_f, \theta_g)$. If the LM is composed of L-layers of transformer blocks (Devlin et al., 2019), the function f is decomposed to multiple functions $f = [f^0, \ldots, f^L]$, where each block gets the output of the previous block as the input: $\boldsymbol{H}^l = f^l(\boldsymbol{H}^{l-1})$.[1]

Knowledge-Augmented Language Model The conventional learning objective defined above might be sufficient for understanding the texts if the tasks require only the general knowledge stored in PLMs. However, it is suboptimal for tackling domain-specific tasks since the general knowledge captured by the parameters θ_f may not include the knowledge required for solving the domain-specific tasks. Thus, contextualizing the texts by the domain knowledge, captured by the domain-specific entities and their relations, is more appropriate for handling such domain-specific problems.

To this end, we propose a function $h(\cdot;\phi)$ which augments PLMs conditioned on the domain knowledge. Formally, the objective for a NLU task with our knowledge-augmented LM is given as follows:

$$\max_{\theta,\phi} \mathcal{L}(\theta,\phi) := \max_{\theta,\phi} \sum_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x};\theta,\phi),$$

$$p(\boldsymbol{y}|\boldsymbol{x};\theta,\phi) = g(\tilde{\boldsymbol{H}};\theta_g),$$

$$\tilde{\boldsymbol{H}}^l = f^l(\boldsymbol{H}^{l-1}, h^l(\boldsymbol{H}^{l-1}, \mathcal{E}, \mathcal{M}, \mathcal{G};\phi);\theta_{f^l}),$$

where ϕ is parameters for the function h, $\mathcal{E}$ is the set of entities, $\mathcal{M}$ is the set of corresponding mentions,

[1] f^0 denotes a word embedding layer which gets $\boldsymbol{x}$ as an input, i.e., $\boldsymbol{H}^0 = f^0(\boldsymbol{x})$, for the sake of simplicity.

and $\mathcal{G}$ is a knowledge graph. In the following, we will describe the definition of the knowledge-related inputs $\mathcal{E}, \mathcal{M}, \mathcal{G}$, and the details of $h(\cdot, \phi)$.

Definition 1 (Entity and Mention). Given a sequence of tokens $\boldsymbol{x} = [w_1, \ldots, w_{|\boldsymbol{x}|}]$, let $\mathcal{E}$ be a set of entities in $\boldsymbol{x}$. Then an **entity** $e \in \mathcal{E}$ is composed of one or multiple adjacent tokens within the input text: $[w_{m^\alpha}, \ldots, w_{m^\omega}] \sqsubseteq \boldsymbol{x}$[2]. Here, $m = (m^\alpha, m^\omega)$ is a **mention** that denotes the start and end locations for the entity within the input tokens $\boldsymbol{x}$, which term is commonly used for defining entities (Févry et al., 2020). Consequently, for each given input $\boldsymbol{x}^{(i)}$, there are a set of entities $\mathcal{E}^{(i)} = \{e_1, \ldots, e_K\}$ and their corresponding mentions $\mathcal{M}^{(i)} = \{m_1, \ldots, m_K\}$. For example, given an input $\boldsymbol{x} = [\text{New, York, is, a, city}]$, we have two entities $\mathcal{E} = \{\textit{New\_York}, \textit{city}\}$ and their associated mentions $\mathcal{M} = \{(1, 2), (4, 4)\}$.

We further construct the entity vocabulary $\mathcal{E}_{\text{train}} = \bigcup_{i=1}^{N} \mathcal{E}^{(i)}$, which consists of all entities appearing in the training dataset. However, at test time, we may encounter unseen entities that are not in $\mathcal{E}_{\text{train}}$. To tackle this, we regard unknown entities as the null entity $e_\emptyset$, so that $\forall e \in \mathcal{E}_{\text{train}} \cup \{e_\emptyset\}$.

Definition 2 (Entity Memory). Given a set of all entities $\mathcal{E}_{\text{train}} \cup \{e_\emptyset\}$, we represent them in the continuous vector (feature) space to learn meaningful entity embeddings. In order to implement this, we define the **entity memory** $\boldsymbol{E} \in \mathbb{R}^{(|\mathcal{E}_{\text{train}}|+1) \times d}$ that comprises of an entity $e \in \mathbb{R}$ as a key and its embedding $\boldsymbol{e} \in \mathbb{R}^d$ as its value. Also, to access the value in the entity memory, we define the point-wise memory access function `EntEmbed` which takes an entity as an input. For instance, $\boldsymbol{e} =$ `EntEmbed`(*New_York*) returns the embedding of the *New_York* entity, and $\boldsymbol{e} =$ `EntEmbed`$(e_\emptyset)$ returns the zero embedding. This entity memory $\boldsymbol{E}$ is the part of the parameter ϕ used in function h.

Definition 3 (Knowledge Graph). Since the entity memory alone cannot represent relational information between entities, we further define a **Knowledge Graph** (KG) $\mathcal{G}$ that consists of a set of factual triplets $\{(h, r, t)\}$, where the head and the tail entities, h and t, are the elements of $\mathcal{E}$, and a relation r is an element of a set of relations $\mathcal{R}$: $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. We assume that a pre-constructed KG $\mathcal{G}^{(i)}$ is **given** for each input $x^{(i)}$, and provide the details of the KGs and how to construct them in **Appendix** A.

[2] $E \sqsubseteq E'$ iff $E = E'$, or E is included in E' such that the order of elements in E and E' is the same.

3.2 Knowledge-conditioned Feature Modulation on Transformer

The remaining problem is how to augment a PLM by conditioning it on the domain-specific knowledge, through the function h. An effective approach to do so without stacking additional layers on top of the LM is to interleave the knowledge from h with the pre-trained parameters of the language model (Devlin et al., 2019) consisting of transformer layers (Vaswani et al., 2017). Before describing our interleaving method in detail, we first describe the Transformer architecture.

Transformer Given $|x|$ token representations $\boldsymbol{H}^{l-1} = [\boldsymbol{h}_1^{l-1}, \ldots, \boldsymbol{h}_{|\boldsymbol{x}|}^{l-1}] \in \mathbb{R}^{|\boldsymbol{x}| \times d}$ from the layer $l-1$ where d is the embedding size, each transformer block outputs the contextualized representations for all tokens. In detail, the l-th block consists of the multi-head self-attention (*Attn*) layer and the residual feed-forward (*FF*) layer as follows:

$$\begin{aligned}
\hat{\boldsymbol{H}}^l &= LN(\boldsymbol{H}^{l-1} + Attn(\boldsymbol{H}^{l-1})) \\
FF(\hat{\boldsymbol{H}}^l) &= \sigma(\hat{\boldsymbol{H}}^l \cdot \boldsymbol{W}_1) \cdot \boldsymbol{W}_2, \\
\boldsymbol{H}^l &= LN(\hat{\boldsymbol{H}}^l + FF(\hat{\boldsymbol{H}}^l)),
\end{aligned} \tag{308}$$

where LN is a layer normalization (Ba et al., 2016), σ is an activation function (Hendrycks and Gimpel, 2016), $\boldsymbol{W}_2 \in \mathbb{R}^{d' \times d}$ and $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d'}$ are weight matrices, and d' is an intermediate hidden size. We omit the bias term for brevity.

Linear Modulation on Transformer An effective yet efficient way to fuse knowledge from different sources without modifying the original model architecture is to scale and shift the features of one source with respect to the data from another source (Dumoulin et al., 2018). This scheme of feature-wise affine transformation is effective on various tasks, such as language-conditioned image reasoning (Perez et al., 2018) or style-transfer in image generation (Huang and Belongie, 2017).

Motivated by them, we propose to linearly transform the intermediate features after the layer normalization of the transformer-based PLM, conditioned on the knowledge sources $\mathcal{E}, \mathcal{M}, \mathcal{G}$. We term this method as the **Knowledge-conditioned Feature Modulation** (KFM), described as follows:

$$\begin{aligned}
\boldsymbol{\Gamma}, \boldsymbol{B}, \tilde{\boldsymbol{\Gamma}}, \tilde{\boldsymbol{B}} &= h^l(\boldsymbol{H}^{l-1}, \mathcal{E}, \mathcal{M}, \mathcal{G}; \phi), \\
\hat{\boldsymbol{H}}^l &= \boldsymbol{\Gamma} \circ LN(\boldsymbol{H}^{l-1} + Attn(\boldsymbol{H}^{l-1})) + \boldsymbol{B}, \\
FF(\hat{\boldsymbol{H}}^l) &= \sigma(\hat{\boldsymbol{H}}^l \cdot \boldsymbol{W}_1) \cdot \boldsymbol{W}_2, \\
\tilde{\boldsymbol{H}}^l &= \tilde{\boldsymbol{\Gamma}} \circ LN(\hat{\boldsymbol{H}}^l + FF(\hat{\boldsymbol{H}}^l)) + \tilde{\boldsymbol{B}}, \qquad (1)
\end{aligned} \tag{330}$$

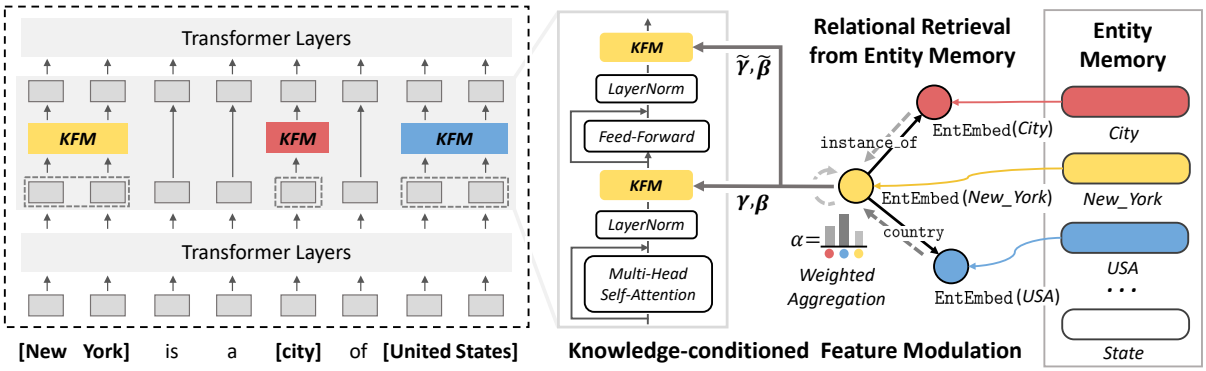


Figure 3: **Framework Overview. (Left)** The architecture of a knowledge-augmented LM with our method. Some of the input tokens are annotated as entities with their mentions. **(Middle)** Inside the transformer block, KFM (§3.2) is applied after the layer normalization as in equation 1, to modulate the hidden representations of tokens in entity mentions. **(Right)** The retrieved embedding of an entity *New_York* is composed by the weighted aggregation of neighbors through the knowledge graph (§3.3).

where $\boldsymbol{H}^{l-1} \in \mathbb{R}^{|x| \times d}$ is the matrix of hidden representations from the previous layer, $\circ$ denotes the hadamard (element-wise) product, and $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{|\boldsymbol{x}|}] \in \mathbb{R}^{|\boldsymbol{x}| \times d}$, $\boldsymbol{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{|\boldsymbol{x}|}] \in \mathbb{R}^{|\boldsymbol{x}| \times d}$. $\boldsymbol{\Gamma}$ and $\boldsymbol{B}$ are learnable modulation parameters from the function h, which are conditioned by the entity representation. For instance, in Figure 3, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ for token 'New' are conditioned on the corresponding entity *New_York*. However, if tokens are not part of any entity (e.g., 'is'), $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ for such tokens are fixed to $\mathbf{1}$ and $\mathbf{0}$, respectively.

One notable advantage of our KFM is that multiple tokens associated to the identical entity are affected by the same modulation (e.g., 'New' and 'York' in Figure 3), which allows the PLM to know which adjacent tokens are in the same entity. This is important for representing the tokens of the domain entity (e.g., 'cod' and 'on'), since the original PLM might regard them as separate, unrelated tokens (See analysis in §5.4 with Figure 5). However, with our KFM, the PLM can identify associated tokens and embed them to be close to each other.

Then, how can we design such functional operations in h? The easiest way is to retrieve the entity embedding of e, associated to the typical token, from the entity memory $\boldsymbol{E}$, and then use the retrieved entity embedding as the input to obtain $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ for every entity (See Figure 3). Formally, for each entity $e \in \mathcal{E}$ and its mention $(m^\alpha, m^\omega) \in \mathcal{M}$,

$$\begin{aligned} \boldsymbol{v} &= \texttt{EntEmbed}(e) \qquad (2) \\ \boldsymbol{\gamma}_j &= \mathbf{1} + h_1(\boldsymbol{v}),\ \boldsymbol{\beta}_j = h_2(\boldsymbol{v}), \\ \tilde{\boldsymbol{\gamma}}_j &= \mathbf{1} + h_3(\boldsymbol{v}),\ \tilde{\boldsymbol{\beta}}_j = h_4(\boldsymbol{v}), \quad m^\alpha \le j \le m^\omega, \end{aligned}$$

where $\boldsymbol{v}$ is the retrieved entity embedding from the entity memory, h_1, h_2, h_3, and h_4 are mutually independent Multi-Layer Perceptrons (MLPs) which return a zero vector $\mathbf{0}$ if $e = e_\emptyset$.

3.3 Relational Retrieval from Entity Memory

Although the simple access to the entity memory can retrieve the necessary entity embeddings for the modulation, this approach has obvious drawbacks as it not only fails to reflect the relations with other entities, but also regards unseen entities as the same null entity $e_\emptyset$. If so, all unseen entities are inevitably modulated by the same parameters even if they have essentially different meaning.

To tackle these limitations, we further consider the relational information between two entities that are linked with a particular relation. For example, the entity *New_York* alone will not give meaningful information. However, with two associated facts (*New_York*, *instance of*, *city*) and (*New_York*, *country*, *USA*), it is clear that *New_York* is a city in the *USA*. Motivated by this observation, we propose **Relational Retrieval** which leverages a KG $\mathcal{G}$ to retrieve entity embeddings from the memory, according to the relations defined in the given KG (See Figure 3, right).

More specifically, our goal is to effectively utilize the relations among entities in $\mathcal{G}$, to improve the `EntEmbed` function in equation 2. We tackle this objective by utilizing a *Graph Neural Network (GNN)* which learns feature representations of each node using a neighborhood aggregation scheme (Hamilton et al., 2017), as follows:

$$\begin{aligned} \boldsymbol{v} = \ & \texttt{UPDATE}(\texttt{EntEmbed}(e), \\ & \texttt{AGG}(\{\texttt{EntEmbed}(\hat{e}) : \forall \hat{e} \in \mathcal{N}(e; \mathcal{G})\})), \end{aligned}$$

where $\mathcal{N}(e; \mathcal{G})$ is a set of neighboring entities of the entity e, `AGG` is the function that aggregates embeddings of neighboring entities of e, and `UPDATE` is the function that updates the representation of e with the aggregated messages from `AGG`.

We consider the attentive scheme (Velickovic et al., 2018; Brody et al., 2021) for neighborhood aggregation, to allocate weights to the target entity's neighbors by their importance. This scheme is helpful in filtering out less useful relations. Formally, we first define a scoring function ψ that calculates a score for every triplet (e_i, r_{ij}, e_j), which is then used to weigh each node during aggregation:

$$\boldsymbol{e}_i = \texttt{EntEmbed}(e_i),\ \boldsymbol{e}_j = \texttt{EntEmbed}(e_j),$$
$$\boldsymbol{e}^* = [\boldsymbol{e}_i \parallel \boldsymbol{r}_{ij} \parallel \boldsymbol{e}_j \parallel \boldsymbol{h}_{e_i}],$$
$$\psi(\boldsymbol{e}_i, \boldsymbol{r}_{ij}, \boldsymbol{e}_j, \boldsymbol{h}_{e_i}) = \boldsymbol{a}^\top \sigma(\boldsymbol{W} \cdot \boldsymbol{e}^*),$$

where σ is a nonlinear activation, $\boldsymbol{e}^* \in \mathbb{R}^{4d}$ is concatenated vector where $\parallel$ denotes the concatenation, $\boldsymbol{a} \in \mathbb{R}^d$ and $\boldsymbol{W} \in \mathbb{R}^{d \times 4d}$ are learnable parameters, $\boldsymbol{r}_{ij} \in \mathbb{R}^d$ is a embedding of the relation, and $\boldsymbol{h}_{e_i} \in \mathbb{R}^d$ is a context representation of the entity e_i obtained from the intermediate hidden states of the LM [3].

The scores obtained from ψ are normalized across all neighbors $e_j \in \mathcal{N}(e_i; \mathcal{G})$ with softmax:

$$\alpha_{ij} = \text{softmax}(\psi(\boldsymbol{e}_i, \boldsymbol{r}_{ij}, \boldsymbol{e}_j)) = \frac{\exp(\psi(\boldsymbol{e}_i, \boldsymbol{r}_{ij}, \boldsymbol{e}_j))}{\sum_{e_{j'} \in \mathcal{N}(e_i; \mathcal{G})} \exp(\psi(\boldsymbol{e}_i, \boldsymbol{r}_{ij'}, \boldsymbol{e}_{j'}))}.$$

Then, we update the entity embedding with a weighted average of the neighboring nodes with α as an attention coefficient, denoted as follows:

$$\boldsymbol{v} = \texttt{UPDATE}\left(\sum_{e_{j'} \in \mathcal{N}(e_i; \mathcal{G})} \alpha_{ij} \cdot \boldsymbol{e}_{j'}\right). \quad (3)$$

By replacing the `EntEmbed` function in equation 2 with the above GNN in equation 3, we now represent each entity with its relational information in KG. This relational retrieval has several advantages over simple retrieval of a single entity from the entity memory. First, the relational retrieval with KG can consider richer interactions among entities, as described in Figure 3.

In addition, we can naturally represent an unseen entity – which is not seen during training but appears at test time – through neighboring aggregation, which is impossible only with the entity memory. In Figure 2, we provide an illustrative example of the unseen entity representation, where the unseen entity *restenosis* is represented with a weighted sum of representations of its neighboring entities *myocardial_infarction*, *asthma*, and *pethidine*, which is beneficial when the set of entities for training and test datasets have small overlaps.

[3] The context representation of the entity is calculated with its mention as follows: $\boldsymbol{h}_e = \frac{1}{m^\omega - m^\alpha + 1} \sum_{i=m^\alpha}^{m^\omega} \boldsymbol{h}_i^{l-1}$

4 Experiment

4.1 Tasks and Datasets

We evaluate our model on two NLU tasks: Question Answering (QA) and Named Entity Recognition (NER). **For QA**, we use three domain-specific datasets: NewsQA (News, Trischler et al., 2017) and two subsets (Relation, Medication) of EMRQA (Clinical, Pampari et al., 2018). We use the Exact-Match (EM) and the F1 score as evaluation metrics. **For NER**, we use three datasets from different domains, namely CoNLL-2003 (News, Sang and Meulder, 2003), WNUT-17 (Social Media, Derczynski et al., 2017) and NCBI-Disease (Biomedical, Dogan et al., 2014). We use the F1 score as the evaluation metric. We report statistics and detailed descriptions of each dataset in **Appendix** B.2.

4.2 Baselines

1. **Vanilla Fine-Tuning (FT)**: A baseline that directly fine-tunes the LM on downstream tasks.
2. **Fine-Tuning +** *more params*: A baseline with one more transformer layer at the end of the LM. We use this baseline to show that the performance gain of our model does not come from the use of additional parameters.
3. **Task-Adaptive Pre-training (TAPT)**: A baseline that further pre-trains the PLM on task-specific corpus as in Gururangan et al. (2020).
4. **TAPT + RecAdam**: A baseline that uses RecAdam (Chen et al., 2020) during further pre-training of PLMs (i.e., TAPT), to alleviate catastrophic forgetting of the learned general knowledge in PLMs from adaptive pre-training.
5. **Domain-Adaptive Pre-training (DAPT)**: A strong baseline that uses a large-scale domain corpus outside the training set during further pre-training (Gururangan et al., 2020), and requires extra data and large computational overhead.
6. **KALA** (pointwise): A variant of KALA that only uses the **entity memory** and does not use the knowledge graphs.
7. **KALA** (relational): Our full model that uses KGs to perform relational retrieval from the entity memory.

4.3 Experimental Setup

We use the uncased BERT-base (Devlin et al., 2019) as the base PLM for all our experiments on QA and NER tasks. For more details on training and implementation, please see the **Appendix** B.

Method	NewsQA	Relation	Medication
Fine-Tuning	53.06 ± 0.63 | 67.20 ± 0.19	54.01 ± 1.14 | 61.43 ± 1.18	12.50 ± 0.28 | 43.31 ± 0.67
+ *more params*	53.59 ± 0.99 | 67.79 ± 0.67	54.06 ± 1.35 | 62.07 ± 1.44	12.46 ± 0.25 | 42.74 ± 0.91
TAPT	53.47 ± 1.69 | 67.59 ± 1.44	53.57 ± 2.05 | 60.87 ± 2.52	12.58 ± 0.42 | 43.82 ± 1.10
+ RecAdam	53.95 ± 1.02 | 67.89 ± 0.75	54.88 ± 1.94 | 62.54 ± 2.14	12.63 ± 0.30 | 43.86 ± 0.87
DAPT†	53.68 ± 0.94 | 67.76 ± 0.61	55.29 ± 1.74 | 62.25 ± 1.80	12.67 ± 0.27 | 43.26 ± 0.88
KALA *(point-wise)*	53.41 ± 0.74 | 67.30 ± 0.45	**56.13** ± 0.85 | **64.69** ± 0.92	12.01 ± 0.47 | 42.97 ± 0.70
KALA *(relational)*	**54.25** ± 0.63 | **68.27** ± 0.63	55.96 ± 1.37 | 64.22 ± 1.15	**12.75** ± 0.61 | **44.19** ± 0.46

Table 1: Experimental results of the extractive QA task on three different datasets with the BERT-base. The reported results are means and standard deviations of performances over five different runs with Exact Match / F1 score as a metric. The numbers in bold fonts denote the best score. † indicates the method under an **extremely high** computational resource setting (See Figure 1).

Method	CoNLL-2003	WNUT-17	NCBI-Disease
Fine-Tuning	90.58 ± 0.19	45.70 ± 1.25	84.42 ± 0.58
+ *more params*	90.75 ± 0.23	46.42 ± 0.55	84.70 ± 0.49
TAPT	90.61 ± 0.73	45.39 ± 0.77	84.39 ± 0.73
+ RecAdam	90.69 ± 0.30	46.73 ± 0.94	84.99 ± 0.88
DAPT†	90.30 ± 0.39	48.29 ± 1.08	84.68 ± 1.63
KALA *(point-wise)*	90.96 ± 0.21	47.33 ± 0.82	85.10 ± 0.73
KALA *(relational)*	**91.02** ± 0.29	**48.35** ± 0.92	**85.77** ± 0.43

Table 2: Experimental results of the NER task on three different datasets with the BERT-base. The reported results are means and standard deviations over five different runs with an F1 score as a metric. The numbers in bold fonts denote the best score. † indicates the baseline under an **extremely high** computational resource setting (See Figure 1).

KFM (§3.2) Components	NewsQA EM	NewsQA F1
None (Fine-tuning)	53.06	67.20
+ $\boldsymbol{\Gamma}, \tilde{\boldsymbol{\Gamma}}$ *(gamma only)*	54.10	67.98
+ $\boldsymbol{B}, \tilde{\boldsymbol{B}}$ *(beta only)*	53.74	67.69
+ $\boldsymbol{\Gamma}, \boldsymbol{B}$ *(first only)*	53.77	67.88
+ $\tilde{\boldsymbol{\Gamma}}, \tilde{\boldsymbol{B}}$ *(second only)*	53.89	67.49
+ $\boldsymbol{\Gamma}, \boldsymbol{B}, \tilde{\boldsymbol{\Gamma}}, \tilde{\boldsymbol{B}}$ **(final)**	**54.25**	**68.27**

Table 3: An ablation study of the KFM parameters $\boldsymbol{\Gamma}$, $\boldsymbol{B}$, $\tilde{\boldsymbol{\Gamma}}$, $\tilde{\boldsymbol{B}}$. We report the average over five different runs.

Architecture Variants (§5.2)	NewsQA EM	NewsQA F1
ERNIE	53.35	67.49
Adapter	53.32	67.38
KT-Net	53.15	67.01
EaE	53.00	67.40
ERICA	51.99	66.40
KALA (ours)	**54.25**	**68.27**

Table 4: Experimental results on knowledge integration architecture variants. We report the average over five different runs.

4.4 Experimental Results

Performance on QA and NER tasks On both extractive QA and NER tasks, our KALA outperforms all baselines, including TAPT and TAPT+RedcAdam (Gururangan et al., 2020; Chen et al., 2020), as shown in Table 1 and 2. These results show that our KALA is highly effective for the language model adaptation task. KALA also largely outperforms DAPT (Gururangan et al., 2020) which is trained with extra data and requires a significantly higher computational cost compare to KALA (See Figure 1 for the plot of efficiency, discussed in Section 5.2).

Effect of Using more Parameters One may suspect whether the performance of our KALA comes from the increment of parameters. However, the experimental results in Table 1 and 2 show that increasing the parameters for PLM during fine-tuning (+ more params) yields marginal performance improvements over naive fine-tuning. This result confirms that the performance improvement of KALA is not due to the increased number of parameters.

Importance of Relational Retrieval The performance gap between KALA (relational) and KALA (point-wise) shows the effectiveness of relational retrieval for language model adaptation, which allows us to incorporate relational knowledge into the PLM. The relational retrieval also helps address unseen entities, as discussed in Section 5.3.

5 Analysis and Discussion

5.1 Ablation Studies

We perform an ablation study to see how much each component of KALA contributes to the performance gain.

KFM Parameters We first analyze the effect of feature modulation parameters (i.e., gamma and beta) in transformers by ablating a subset of them in Table 3, in which we observe that using both gamma and beta after both layer normalization on a transformer layer obtains the best performance.

Architectural Variants We now examine the effectiveness of the proposed knowledge conditioning scheme in our KALA framework. To this end, we use or adapt the knowledge integration methods from previous literature, to compare their effectiveness. Specifically, we couple the following five components with KALA: Entity-as-Experts (Févry et al., 2020), Adapter (Houlsby et al., 2019), KT-Net (Yang et al., 2019), ERNIE (Zhang et al., 2019), and ERICA (Qin et al., 2021). Note that, most of them were proposed for improving pre-training from scratch, while we adapt them for fine-tuning under our KALA framework (The details are given in **Appendix** B.4). As shown in Table 4, our KFM used in KALA outperforms all variants, demonstrating the effectiveness of feature modulation in the middle of transformer layers for fine-tuning.

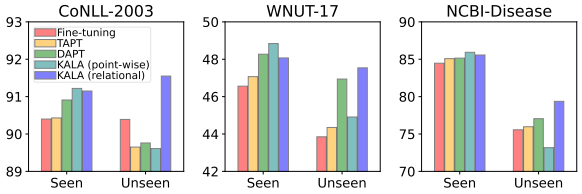


Figure 4: Results on seen and unseen, where Seen denotes the context having less than 3 unseen entities, otherwise Unseen.

5.2 Efficiency

Figure 1 illustrates the performance and training FLOPs of KALA against baselines on the NewsQA dataset. We observe that the performance of TAPT decreases with the increased number of iterations, which could be due to forgetting of the knowledge from the PLM. DAPT, while not suffering from performance loss, requires huge computational costs as it trains on 112 times larger data for further pre-training (See **Appendix** B.3 for detail). On the other hand, our KALA outperforms DAPT without using external data, while requiring 17 times fewer computational costs, which shows that KALA is not only effective but also highly efficient. The resource requirement of our KALA could be further reduced by adjusting the size of entity memory (e.g., removing less frequent entities).

5.3 Effectiveness on Unseen Entities

One remarkable advantage of our KALA is its ability to represent an unseen entity by aggregating features of its neighbors from a given KG. To analyze this, we first divide all contexts into one of Seen and Unseen, where Seen denotes the context with less than 3 unseen entities, and then measure the performance on the two subsets. As shown in Figure 4, we observe that the performance gain of KALA over the baselines is much larger on the Unseen subset, which demonstrates the effectiveness of KALA's relational retrieval scheme to represent unseen entities. DAPT also largely outperforms fine-tuning and TAPT as it is trained on an extremely large external corpus for adaptive pre-training. However, KALA even outperforms DAPT in most cases, verifying that our knowledge-augmentation method is more effective for tackling domain-specific tasks. The visualization of embeddings of seen and unseen entities in Figure 2 shows that KALA embeds the unseen entities more closely to the seen entities[4], which explains KALA's good performance on the Unseen subset.

[4]We quantitatively measure the mean of cosine distance of each unseen entity to its nearest seen entity, observing that KALA embeds unseen 1.5 times more closer to seen than DAPT (i.e., 0.07 for KALA vs 0.11 for DAPT for distance).

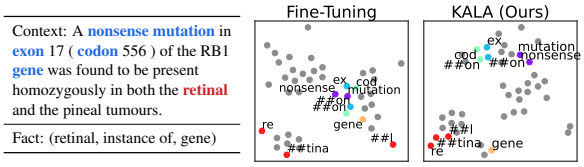


Figure 5: A case study on one context of the NCBI-Disease dataset. A left table shows the context and its fact, and a right figure shows a visualization of token representations. Text in blue and red denote the seen and unseen entities, respectively.

5.4 Case Study

To better see how our KFM (§3.2) works, we show the context and its fact, and then visualize representations from the PLM modulated by the KFM. As shown in Figure 5 right, the token '##on' is not aligned with their corresponding tokens, such as 'ex' (for *exon*) and 'cod' (for *codon*), in the baseline. However, with our feature modulation that transforms multiple tokens associated with the single entity equally, the two tokens (e.g., ('ex', '##on')), composing one entity, are closely embedded. Also, while the baseline cannot handle the unseen entity consisting of three tokens: 're', '##tina', and '##l', KALA embeds them closely by representing the unseen *retinal* from the representation of its neighborhood *gene* derived by the domain knowledge – (*retinal*, *instance of*, *gene*).

5.5 Extension to Generative Model

Our KALA framework is also applicable to encoder-decoder PLMs by applying the KFM to the encoder. We further validate KALA's effectiveness on the encoder-decoder PLMs on the generative QA task (Lee et al., 2021) with T5-small (Raffel et al., 2020). Table 5 shows that KALA largely outperforms baselines even with such a generative PLM.

T5-small	NewsQA EM	NewsQA F1
Fine-tuning	48.96	64.24
TAPT	48.66	64.30
+ RecAdam	48.37	63.41
KALA (ours)	**51.78**	**66.88**

Table 5: Results of generative QA.

6 Conclusion

In this paper, we introduced KALA, a novel framework for language model adaptation, which modulates the intermediate representations of a PLM by conditioning it with the entity memory and the relational facts from KGs. We validated KALA on various domains of QA and NER tasks, on which KALA significantly outperforms relevant baselines while being computationally efficient. We demonstrate that the success of KALA comes from both KFM and relational retrieval, allowing the PLM to recognize entities but also handle unseen ones that might frequently appear in domain-specific tasks.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint*, arXiv:1607.06450.

Fan Bai, Alan Ritter, and Wei Xu. 2021. Pre-train or annotate? domain adaptation with a constrained budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5002–5015.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3613–3618.

Shaked Brody, Uri Alon, and Eran Yahav. 2021. How attentive are graph attention networks? *arXiv preprint*, arXiv:2105.14491.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7870–7881.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 140–147. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *J. Biomed. Informatics*, 47:1–10.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. 2018. Feature-wise transformations. *Distill*.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4937–4951.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2021. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint*, arXiv:2108.05036.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 4237–4247.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint*, arXiv:1606.08415.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Xun Huang and Serge J. Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1510–1519. IEEE Computer Society.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *arXiv preprint*, arXiv:2009.13081.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240.

Seanie Lee, Minki Kang, Juho Lee, and Sung Ju Hwang. 2021. Learning to perturb word embeddings for out-of-distribution QA. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5583–5595.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint*, arXiv:1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814.

Anusri Pampari, Preethi Raghavan, Jennifer J. Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2357–2368.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 43–54.

Yujia Qin, Yankai Lin, Ryuichi Takanobu, Zhiyuan Liu, Peng Li, Heng Ji, Minlie Huang, Maosong Sun, and Jie Zhou. 2021. ERICA: improving entity and relation understanding for pre-trained language models via contrastive learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing,*

ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 3350–3363.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4603–4611.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W. Cohen. 2021. Adaptable and interpretable neural memoryover symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3678–3691.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6442–6454.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2346–2357.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 764–777.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3320–3328.

Xiang Yue, Bernal Jimenez Gutierrez, and Huan Sun. 2020. Clinical reading comprehension: A thorough analysis of the emrqa dataset. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4474–4486.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451.

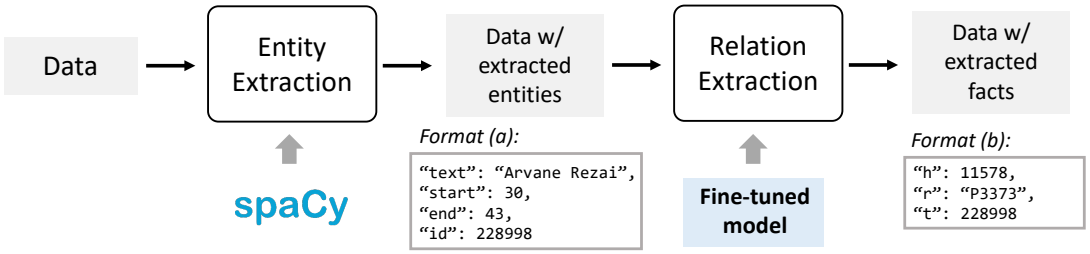


Figure 6: Visual diagram of the KG construction pipeline used in this work. The entity format is composed of its corresponding text in the data, its character-level mention boundary, and its wikidata id. The fact format is composed of the head, relation, and tail, where head and tail entities are represented with their wikidata ids following the entity format.

Hyperparameters	NewsQA	Relation	Medication	CoNLL-2003	WNUT-17	NCBI-Disease
LM for Relation Extraction	BERT-base-uncased					
Threshold on Relation Extraction	0.1					
Size of Entity Memory	62823	5724	4635	10288	101	3502
The location of KFM	11	11	11	8	9, 11	8, 10

Table 6: **Hyperparamters for Knowledge Graph (Top) and KALA (Bottom)** on six datasets we used. The reported performances on main paper are measured with the above settings.

A Details on KG Construction

In this work, we propose to use the Knowledge Graph (KG) that can define the relational information among entities that only appear in each dataset. However, unfortunately, most of the task datasets do not contain such relational facts on its context, thus we need to construct them manually to obtain the knowledge graph. In this section, we explain the way of constructing the knowledge graph that we used, consisting of facts of entities for each context in the task dataset.

Relation extraction is the way how we obtain the factual knowledge from the text of the target dataset. To do so, we first need to extract entities and their corresponding mentions from the text, and then link it to the existing entities in wikidata (Vrandecic and Krötzsch, 2014). In order to do this, we use the existing library named as spaCy[5], and open-sourced implementation of Entity Linker[6]. To sum up, in our work, a set of entities $\mathcal{E}^{(i)}$ and corresponding mentions $\mathcal{M}^{(i)}$ for the given input $\boldsymbol{x}^{(i)}$ are obtained through this step. Regarding a concrete example, please see format (a) in Figure 6. In the example, "Text" indicates the entity mention within the input $\boldsymbol{x}$, the "start" and "end" indicates its mention position denoted as (m^{α}, m^{ω}), and "id" indicates the wikidata id for the entity identification used in the next step.

To extract the relation among entities that we obtained above, we use the scheme of Relation Extraction (RE). In other words, we use the trained RE model to build our own knowledge base (KB) instead of using the existing KG directly from the existing general-domain KB[7]. Specifically, we first fine-tune the BERT-base model (Devlin et al., 2019) for 2 epochs with 600k distantly supervised data used in Qin et al. (2021), where the Wikipedia document and the Wikidata triplets are aligned. Then, we use the fine-tuned BERT model to extract the relations between entity pairs in the text. We use the model with a simple bilinear layer on top of it, which is widely used scheme in the relation extraction literature (Yao et al., 2019). For an example of the extracted fact, please see format (b) in Figure 6. In the example, "h" denotes the wikidata id of the head entity, "r" denotes the wikidata id of the extracted relation, and "t" denotes the wikidata id of the tail entity. In the relation extraction, the model returns the categorical distribution over the top 100 frequent relations. In general, the relation of top-1 probability is used as the relation for the corresponding entity pair. However, this approach sometimes results in predicting `no_relation` on most entity pairs. Thus, to obtain more relations, we further use the relation of top-2 probability in the case where `no_relation` has a top-1

[5]https://spacy.io/

[6]https://github.com/egerber/spaCy-entity-linker

[7]We faced several problems here. First of all, most KBs such as Wikidata are less informative, especially for the entities included in the domain-specific context (e.g., News, Medical records). It only has a few facts for each context of domain-specific tasks, although we can find a lot of entities included in the context. Second, the entity linker is imperfect. Due to the wrongly linked entity to the wikidata, even existing relations in the KG are ignored a lot. Therefore, we instead use a trained neural network to effectively extract the relations between entities, instead of direct querying to obtain facts.

Dataset	Training			Validation			Test		
	# Context	C. Length	# Question	# Context	C. Length	# Question	# Context	C. Length	# Question
NewsQA	11428	655.7	74160	-	-	-	106	625.8	674
Relation	296	1386.1	6162	42	1206.6	321	85	1467.7	802
Medication	182	1737.3	7518	26	1626.5	1858	53	2005.0	4005

Table 7: **QA dataset statistics.** We report the number of contexts and questions (i.e., # Context and # Question), with the average length of contexts (i.e., C. Length) where the length is measured as the number of tokens after wordpiece tokenization.

probability but the top-2 probability is larger than a certain threshold (e.g., > 0.1). In Figure 6, we summarize our KG construction pipeline. In Table 6, we report the hyperparameters related to our KG construction.

B Experimental Setup

In this section, we introduce the detailed setups for our models and baselines used in Table 1, 2, and 4.

B.1 Implementation Details

We use the Pytorch (Paszke et al., 2019) for the implementation of all models. Also, to easily implement the language model, we use the huggingface library (Wolf et al., 2020) containing various transformer-based pre-trained language models (PLMs) and their checkpoints.

Details for KALA In this paragraph, we describe the implementation details of the components, such as four linear layers in the proposed KFM, architectural specifications in the attention-based GNN, and initialization of both the entity memory and relational embeddings, in the following. In terms of the functions h_1, h_2, h_3, and h_4 in the KFM of Equation 2, we use two linear layers with the ReLU (Nair and Hinton, 2010) activation function, where the dimension is set to 768.

For relational retrieval, we implement the novel GNN model based on GATv2 (Brody et al., 2021) provided by the torch-geometric package (Fey and Lenssen, 2019). Specifically, we stack two GNN layers with the RELU activation function and also use the dropout with a probability of 0.1. For attention in our GNN, we mask the nodes of the null entity, so that the attention score becomes zero for them. Moreover, to obtain the context representation of the entity (See Footnote 3 in the main paper) used in the GNN attention, we use the scatter operation[8] for reduced computational cost.

For Entity Memory, we experimentally found that initializing the embeddings of the entity memory with the contextualized features obtained from the pre-trained language model could be helpful. Therefore, the dimension of the entity embedding is set to the same as the language model $d = 768$. For relation embeddings, we randomly initialize them, where the dimension size is set to 128.

Location of KLM in the PLM Note that, the number and location of the KFM layers inside the PLM are hyperparameters. However, we empirically found that inserting one to three KFM layers at the end of the PLM (i.e., after the 9th - 11th layers of the BERT-base language model) is beneficial to the performance (See Appendix C.4 for experiments on diverse layer locations).

B.2 Dataset Details

Here we describe the dataset details with its statistics for two different tasks: extractive question answering (QA) and named entity recognition (NER).

Question Answering We evaluate models on three domain-specific datasets: NewsQA, Relation, and Medication. Notably, NewsQA (Trischler et al., 2017) is curated from CNN news articles. Relation and Medication are originally part of the emrQA (Pampari et al., 2018), which is an automatically constructed question answering dataset based on the electrical medical record from n2c2 challenges[9]. However, Yue et al. (2020) extract two major subsets by dividing the entire dataset into Relation and Medication and suggest the usage of sampled questions from the original emrQA dataset. Following the suggestion of Yue et al. (2020), we use only 1% of generated questions of Relation for training, validation, and testing. Also, we only use 1% of generated questions of Medication for training and use 5% of generated questions of Medication for validation and testing. Since the original emrQA is automatically generated based on templates, the quality is poor – it means that the original emrQA dataset was inappropriate to evaluate the ability of the model to reason over the clinical text since the most of questions can be

[8] https://github.com/rusty1s/pytorch_scatter

[9] https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/

Hyperparameters	NewsQA	Relation	Medication	CoNLL-2003	WNUT-17	NCBI-Disease	Generative NewsQA
Fine-tuning							
Language Model	BERT-base-uncased						T5-small
Maximum Sequence Length	384	384	384	128	128	128	512
Batch Size	12	12	12	32	32	32	64
Training Epochs	2	2	2	20	20	20	4
Optimizer	AdamW						Adafactor
Learning rate	3e-5	3e-5	3e-5	5e-5	5e-5	5e-5	1e-4
Weight Decay	0.01	0.01	0.01	0	0	0	-
LR decay Warmup rate	0.06	0.06	0.06	0	0	0	-
Half Precision	Yes	Yes	Yes	No	No	No	No
Task-Adaptive Pre-training (TAPT)							
Maximum Sequence Length	384	384	384	128	128	128	384
Batch Size	12	12	12	32	32	32	64
Training Epochs	1	1	1	3	3	3	4
Training Epochs (RecAdam)	3	1	1	3	3	3	4
Optimizer	AdamW						Adafactor
Learning rate	5e-5						1e-3
Weight Decay	0.01	0.01	0.01	0	0	0	-
LR decay Warmup rate	0.06	0.06	0.06	0	0	0	-
Half Precision	Yes	Yes	Yes	No	No	No	No

Table 8: **Hyperparamters for Fine-tuning (Top) and TAPT (Bottom)** on six datasets (+ generative QA) we used for reporting the performances in the main paper. Note that the Fine-tuning setup is applied to all methods including KALA.

	Training		Validation		Test	
Dataset	# Context	C. Length	# Context	C. Length	# Context	C. Length
CoNLL-2003	14,041	19.95	3,250	21.36	3,453	18.77
WNUT-17	3,394	31.32	1,009	19.28	1,287	30.58
NCBI-Disease	5,433	34.36	924	35.00	941	35.50

Table 9: **NER dataset statistics.** We report the number of contexts (i.e., # Context), with the average length of them (i.e., C. Length) on training, validation, and test sets.

answered by the simple text matching. To overcome this limitation, Yue et al. (2020) suggests two ways to make the task more difficult. First, they divide the question templates into easy and hard versions and then use the hard question only. Second, they suggest replacing medical terminologies in the question of the test set into synonyms to avoid the trivial question which can be solvable with a simple text matching. We use both methods to Relation and Medication datasets to report the performance of every model. For more details on Relation and Medication datasets, please refer to the original paper (Yue et al., 2020). The statistics of training, validation, and test sets on all QA datasets are provided in Table 7.

Named Entity Recognition We use three different domain-specific datasets for evaluating our KALA on NER tasks: CoNLL-2003 (Sang and Meulder, 2003) (News), WNUT-17 (Derczynski et al., 2017) (Social Network Service) and NCBI-Disease (Dogan et al., 2014) (Biomedical). The CoNLL-2003 is constructed from the manually curated 1,393 English news articles, including 301.4k tokens, which has 9 class labels. The WNUT-17 dataset consists of 65,124 emerging and rare entities from social media (e.g., Twitter, Reddit, YouTube, to name a few), which has 13 class labels. The NCBI-Disease dataset consists of the 793 PubMed articles from the biomedical domain, which contains 6,892 disease mentions and 790 disease concepts, and also has 3 class labels. The statistics of training, validation, and test sets are provided in Table 9.

Hyperparameters	News	Medical Textbook
Domain-Adaptive Pre-training (DAPT)		
The number of text (by lines)	10M	100k
The number of text (by words)	618M	12.8M
The size of data (by volume)	3.5G	86M
Maximum Sequence Length	384	
Batch Size	64	
Training Epochs	50	
Maximum Steps	12.5k	
Optimizer	AdamW	
Learning rate	5e-5	
Weight Decay	0.01	
LR decay Warmup rate	0.06	
Half Precision	Yes	
Applied Dataset	NewsQA CoNLL-2003 WNUT-17	Relation Medication NCBI-Disease

Table 10: **Hyperparamters for DAPT** on two domains we used for reporting the performances in the main paper.

B.3 Training details

All experiments are constrained to be done with a single 12GB Geforce RTX 2080 Ti GPU for fairness in terms of memory and the availability on the academic budget, except for the DAPT and generative QA which use a single 48GB Quadro 8000 GPU. KALA training needs 3 hours in wall time with a single GPU. For all experiments, we select the best checkpoint on the validation set. For the summary of training setups, please see Table 8 and 10.

Fine-tuning Setup In the following three paragraphs, we explain the setting of fine-tuning for QA, NER, and generative QA tasks. For all experiments on extractive QA tasks, we fine-tune the Pre-trained Language Model (PLM) for 2 epochs with the weight decay of 0.01, learning rate of 3e-5, maximum sequence length of 384, batch size of 12, linear learning rate decay of 0.06 warmup rate, and half precision (Micikevicius et al., 2018).

For all experiments on NER tasks, we fine-tune the PLM for 20 epochs, where the learning rate is set to 5e-5, maximum sequence length is set to 128, and batch size is set to 32. We use AdamW (Loshchilov and Hutter, 2019) as an optimizer using BERT-base as the PLM.

For the generative QA task in Table 5, we fine-tune the T5-small (Raffel et al., 2020) for 4 epochs with the learning rate of 1e-4, maximum sequence length of 512, and batch size of 64. We also use the Adafactor (Shazeer and Stern, 2018) optimizer. Instead of training with the same optimizer as in BERT for QA and NER, we instead use the independent AdamW optimizer with the learning rate of 1e-4 and weight decay of 0.01 to train the KALA module with T5.

Adaptive Pre-training Setup In this paragraph, we describe the experimental settings of adaptive pre-training baselines, namely TAPT, TAPT (+ RecAdam), and DAPT. For QA tasks, we further pre-train the PLM for {1,3,5,10} epochs and then report the best performance among them. Specifically, reported TAPT result on NewsQA, Relation, and Medication are obtained by 1 epoch of further pre-training. We use the weight decay of 0.01, learning rate of 5e-5, maximum sequence length of 384, batch size of 12, and linear learning rate decay of 0.06 warmup rate, with a half-precision. Also, the masking ratio for the pre-training objective is set to 0.15, following the existing strategy introduced in the original BERT paper (Devlin et al., 2019).

For NER tasks, we further pre-train the PLM for 3 epochs across all datasets. In particular, the learning rate is set to 5e-5, batch size is set to 32, and the maximum sequence length is set to 128. We also use AdamW (Loshchilov and Hutter, 2019) as the optimizer for all experiments.

In the case of T5-small for generative QA in Table 5, we further pre-train the PLM for 4 epochs with the learning rate of 0.001, batch size of 64, maximum sequence length of 384, and Adafactor (Shazeer and Stern, 2018) optimizer.

Regarding the setting of TAPT (+ RecAdam) on all tasks, we follow the best setting in the original paper (Chen et al., 2020) – sigmoid as an annealing function with annealing parameters: $k = 0.5$, $t_0 = 250$, and the pretraining coefficient of 5000.

For training with DAPT, we need an external corpus having a large amount of data for adaptive pre-training. Thus, we first choose the datasets of two domains – News and Medical. Specifically, as the source of corpus for the News domain, we use the sampled set of 10 million News from the RealNews dataset used in Gururangan et al. (2021). As the source of corpus for the Medical domain, we use the set of approximately 100k passages from the Medical textbook provided in Jin et al. (2020). The size of pre-training data used in DAPT is much larger than TAPT. In other words, for experiments on NewsQA, TAPT only uses fine-tuning contexts containing 5.8 million words from the NewsQA training dataset, while DAPT uses more than a hundred times larger data – enormous contexts containing about 618 million words from the RealNews database. For both News and Medical domains, we further pre-train the BERT-base model for 50 epochs with the batch size of 64, to match the similar computational cost used in Gururangan et al. (2020). Other experimental details are the same as TAPT described above.

B.4 Architectural Variant Details

In this subsection, we describe the details of architectural variants reported in Section 5.1. For all variants, we use the same KGs used in KALA.

Entity-as-Experts (Févry et al. (2020); EaE) utilizes the entity memory similar to our work, but they use the parametric dense retrieval more like the memory neural network (Sukhbaatar et al., 2015). Similar to Févry et al. (2020); Verga et al. (2021), we change the formulation of query and memory retrieval by using the mention representation of the entity from the intermediate hidden states of PLMs, which is formally defined as follows:

$$\boldsymbol{h}_e = \frac{1}{m^\omega - m^\alpha + 1} \sum_{i=m^\alpha}^{m^\omega} \boldsymbol{h}_i^{l-1}, \qquad (4)$$

$$\boldsymbol{v} = \mathrm{softmax}(\boldsymbol{h}_e \cdot \boldsymbol{E}^\top) \cdot \boldsymbol{E},$$

where $\boldsymbol{h}_e$ represents the average of token representations of the entity mention $m = (m^\alpha, m^\omega)$. We also give the supervised retrieval loss (ELLoss

in Févry et al. (2020)), when training the EaE model. With this retrieval, EaE also can represent the unseen entity $e \notin \mathcal{E}_{train}$ if we know the mention boundary of the given entity on the context. We believe it is expected to work well, if the entity memory is pre-trained on the enormous text along with the pre-training of the language model from the scratch. However, it might underperform for the language model adaptation scenario, since it can fall into the problem of circular reasoning – the PLM does not properly represent the unseen entity, but it should predict which entity it is similar from the representation. Regarding the integration of the knowledge from the entity memory into the PLM, the retrieved entity representation $\boldsymbol{v}$ is simply added (Peters et al., 2019) to the hidden representations $\boldsymbol{H}$ after the transformer block as follows:

$$\tilde{\boldsymbol{H}}^l = \boldsymbol{H}^l + h(\boldsymbol{v}) \tag{5}$$

where h is Multi-Layer Perceptrons (MLPs).

Adapter (Houlsby et al., 2019) is introduced to fine-tune the PLM only with a few trainable parameters, instead of fine-tuning the whole parameters of the PLM. To adapt this original implementation into our KALA framework, we replace our Knowledge-conditioned Feature Modulation with it, where the Adapter is used as the knowledge integration module. We interleave the layer of Adapter after the feed-forward layer (FF) and before the residual connection of the transformer block. Also, instead of only providing the LM hidden states as an input, we concatenate the knowledge representation in Equation 3 to the LM hidden states. Note that we fine-tune the whole parameters following our KALA setting, unlike fine-tuning the parameters of only Adapter layers in Houlsby et al. (2019).

ERNIE (Zhang et al., 2019) is a notable PLM model that utilizes the external KB as an input for the language model. The key feature of ERNIE can be summarized into two folds. First, they use the multi-head self-attention scheme (Vaswani et al., 2017) to contextualize the input entities. Second, ERNIE fuses the entity representation at the end of the PLM by adding it to the corresponding language representation. We assume that those two features are important points of ERNIE. Therefore, instead of using a Graph Neural Network (GNN) layer, we use a multi-head self-attention layer to contextualize the entity embeddings. Then, we add it to a representation of the entity from the PLM, which is the same as the design in equation 5.

KT-Net (Yang et al., 2019) uses knowledge as an external input in the fine-tuning stage for extractive QA. Since they have a typical layer for integrating existing KB (Miller, 1995; Carlson et al., 2010) with the PLM, we only adopt the self-matching layer as the architecture variant of the KFM layer used in our KALA framework. The computation of the self-matching matrix in KT-Net is costly, i.e., it requires a large computational cost that is approximately 12 times larger than KALA.

ERICA (Qin et al., 2021) uses contrastive learning in LM pre-training to reflect the relational knowledge into the language model. We use the Entity Discrimination task from ERICA on the primary task of fine-tuning. We would like to note that, as reported in Section 5 of the original paper (Qin et al., 2021), the use of ERICA on fine-tuning has no effect, since the size and diversity of entities and relations in downstream training data are limited. Such limited information rather harms the performance, as it can hinder the generalization. In other words, contrastive learning cannot reflect the entity and relation in the test dataset.

B.5 FLOPs Computation

In this subsection, we give detailed descriptions of how the FLOPs in Figure 1 are measured. We majorly follow the script from the ELECTRA (Clark et al., 2020) repository to compute the approximated FLOPs for all models including ours. For FLOPs computation of our KALA, we additionally include the FLOPs of the entity embedding layer, linear layers for h_1, h_2, h_3, h_4, and GNN layer. Since the GNN layer is implemented based on the sparse implementation, we first calculate the FLOPs of the message propagation over one edge, and then multiply it to the average number of edges per node. Also, in terms of the computation on mentions, we consider the maximum sequence length of the context rather than the average number of mentions, to set the upper bound of FLOPs for our KALA. Note that, in NewsQA training data, the average number of nodes is 57, the average number of edges for each node is 0.64, and the average number of mentions in the context is 92.68.

C Additional Experimental Results

In this section, we provide the analyses on the forgetting of TAPT, entity memory, number of entities and facts, location of the KLM layer, and values of Gamma and Beta.

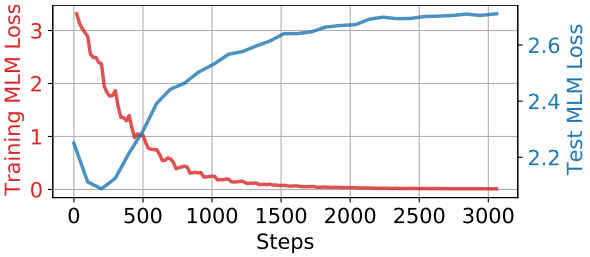


Figure 7: Masked Language Model loss from Task-Adaptive Pre-Training on the domain-specific training dataset (Relation) and the general domain test dataset (Sampled wikipedia).

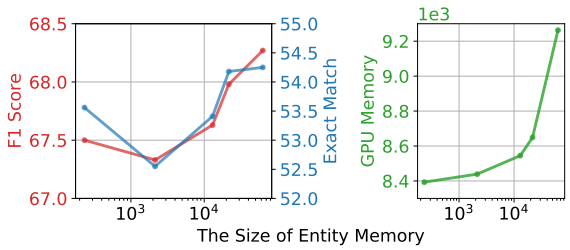


Figure 8: The performance (F1 score and Exact Match) and the GPU memory usage on NewsQA dataset with varying the size of elements in the entity memory.

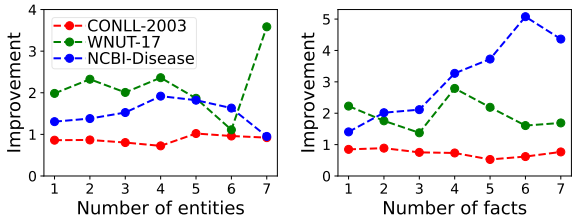


Figure 9: Performance improvements of our KALA from simple fine-tuning, with varying the number of entities and facts in the context on Named Entity Recognition tasks.

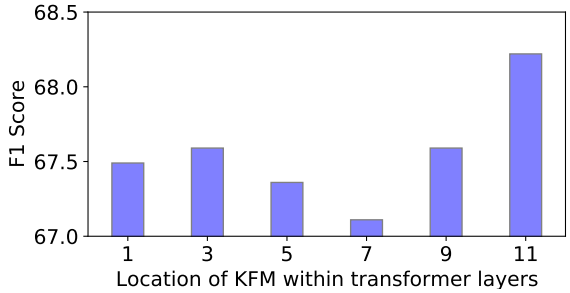


Figure 10: The performance of our KALA with varying the location of the KFM layer inside the BERT-base model. y-axis denotes the F1 score on NewsQA and x-axis denotes the location of the KFM layer. For instance, 11 means the case where the KFM layer is appended in the 11th transformer layer of BERT-base.

C.1 Analysis on forgetting of TAPT

In Figure 1, we observe that the performance of TAPT decreases as the number of training steps increases. To get a concrete intuition on this particular phenomenon, we analysis what happens in the Pre-trained Language Model (PLM), when we further pre-train it on the task-specific corpus. Specifically, in Figure 7, we visualize the Masked Language Model (MLM) loss of TAPT on both domain-specific corpus from the Relation dataset and general corpus from the sampled Wikipedia documents during the adaptive pre-traing. As Figure 7 shows, the test MLM loss increases while the training MLM loss persistently increases as the training step increases. This result indicates that TAPT on domain-specific corpus may yield the catastrophic forgetting of the general knowledge in the PLM.

C.2 Effects of the Size of Entity Memory

In this subsection, we analyze how the size of entity memory affects the performance of our KALA. In Figure 8, we plot the performance of KALA on the NewsQA dataset by varying the number of entity elements in the memory. Note that, we reduce the size of the entity memory by eliminating the entity appearing fewer times. Thus, the results are obtained by only considering the entities that appear more than $[1000, 100, 10, 5, 0]$ times, e.g., 0 means the model with full entity memory. As shown in Figure 8, we observe that the size of the entity memory is larger, the performance of our KALA is better in general. However, interestingly, we also observe that the smallest size of the entity memory shows decent performance, which might be due to the fact that some parameters in the entity memory are stale. For more discussions on it including visualization, please refer to Appendix D.2. Finally, we would like to note that, in Figure 1, we report the performance of our KALA in the case of $[1000, 5, 0]$ (i.e., considering entities appearing more than $[1000, 5, 0]$ times).

C.3 Effects of the Number of Entity and Fact

In this subsection, we aim to analyze which numbers of entities and facts per context are appropriate to achieve good performance in NER tasks. Specifically, we first collect the contexts having more than or equal to the k number of entities (or facts), and then calculate the performance difference from our KALA to the fine-tuning baseline. As shown in Figure 9, while there are no obvious patterns, performance improvements from the baseline are consistent across a varying number of entities and facts. This result suggests that our KALA is indeed beneficial when entities and facts are given to the model, whereas the appropriate number of entities and facts to obtain the best performance against the baseline is different across datasets.

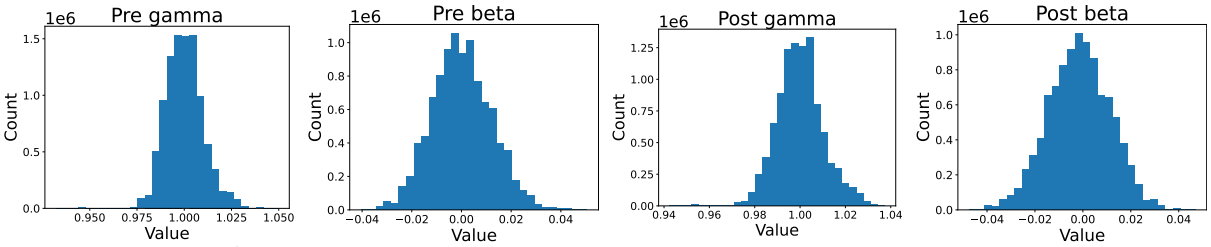


Figure 11: Histogram of values of gamma and beta on the CoNLL-2003 dataset.

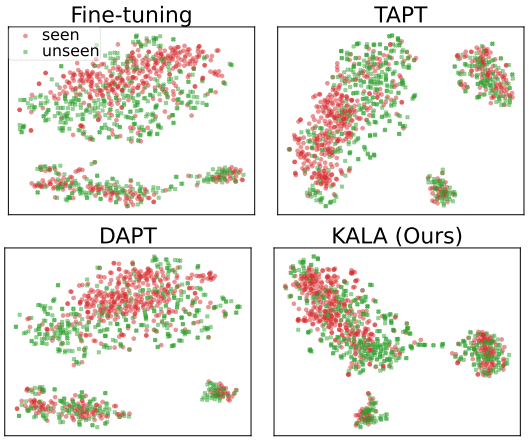


Figure 12: Visualization of contextual representations for seen and unseen entities on the NCBI-Disease dataset.

Method	GPU Mem.	Wall Time	FLOPs (10^{16})
Fine-Tuning	8 GB	3 hrs	9.5
+ *more params*	8.8 GB	3 hrs	10.1
TAPT	8 GB	3.8 hrs	10.9
DAPT	48 GB	40 hrs <	157.0
KALA (ours, 0.2k)	8.2 GB	3 hrs	9.6
KALA (ours, 62.8k)	9.1 GB	3 hrs	10.2

Table 11: Efficiency comparison of GPU memory, Wall Time, and FLOPs on the NewsQA dataset. The number 0.2k and 62.8k indicate the size of entity memory used in our KALA.

C.4 Effects of the Location of KFM

In the main paper and Appendix B.1, we describe that the location of the KFM layer inside the PLM architecture is the hyperparameter. However, someone might wonder which location of KFM yields the best performance, and what is the reason for this. Therefore, in this section, we analyze where we obtain the best performance in various locations of the KFM layer on the NewsQA dataset. Specifically, in Figure 10, we show the performance of our KALA with varying the location of the KFM layer insider the BERT-base model. The results demonstrate that the model with the KFM on the last layer of the BERT-base outperforms all the other choices. This might be because, as the final layer of the PLM is generally considered as the most task-specific layer, our KFM interleaved in the latest layer of BERT expressively injects the task-specific information from the entity memory and KGs, to such a task-specific layer.

C.5 Analysis on Values of Gamma and Beta

To see how much amount of value on gamma and beta is used to shift and scale the intermediate hidden representations in transformer layers, we visualize the modulation values, namely gamma and beta, in Figure 11. We first observe that, as shown in Figure 11, the distribution of values of gamma and beta approximately follow the Gaussian distribution, with zero mean for beta and one mean for gamma. Also, we notice that the scale of values remain nearly around the mean point, which suggests that the small amount of shifting to intermediate hidden representations on transformer layers is enough to contribute to the performance gain, as we can see in the main results of Table 1, 2.

C.6 Detailed Efficiency Comparison

While we provide the efficiency on FLOPs in Figure 1, we further provide the efficiency on GPU memory, wall time, and FLOPs for training each method in Table 11. Specifically, we measure the computational cost on the NewsQA dataset with BERT-base, where we use the single Geforce RTX 2080 Ti GPU on the same machine. For our KALA, as we can flexibly manage the cost of GPU memory by reducing the number of entities in entity memory (See Figure 8 with Appendix C.2 for more analysis on the effects of the size of entity memory), we provide the experimental results on two settings – KALA with memory size 0.2k and 62.8k (full memory). As shown in Table 11, we confirm that the computational cost of our KALA with the full memory is similar to the cost of the *more params* baseline that uses one additional transformer layer on top of BERT-base. However, by reducing the number of entities in the memory, we can achieve better efficiency than more params in terms of GPU memory and FLOPs. Also, we observe that the training cost (i.e., Wall Time and FLOPs) of TAPT and DAPT is high, especially on DAPT, thus we verify that our KALA is more efficient to train for domain adaptation settings.

Method	NewsQA	Relation	WNUT-17	NCBI-Disease
Fine-Tuning	57.21 ± 0.56 | 71.91 ± 0.35	46.61 ± 2.75 | 53.89 ± 2.92	55.00 ± 1.66	86.91 ± 1.08
+ *more params*	**58.07** ± 1.19 | 72.38 ± 1.04	45.12 ± 0.86 | 53.22 ± 1.27	56.62 ± 0.26	87.21 ± 0.26
TAPT	57.24 ± 0.53 | 71.77 ± 0.34	45.66 ± 2.20 | 53.23 ± 2.38	55.46 ± 1.90	86.24 ± 0.76
KALA *(relational)*	58.01 ± 0.57 | **72.70** ± 0.25	**47.40** ± 1.67 | **55.13** ± 1.26	**56.96** ± 0.27	**87.72** ± 0.27

Table 12: Experimental results of the extractive QA and NER tasks on four different datasets – NewsQA, Relation, WNUT-17 and NCBI-Disease – with the RoBERTa-base. The reported results are means and standard deviations over five different runs. We use Exact Match and F1 score as a metric for QA, and F1 score for NER. The numbers in bold fonts denote the best score.

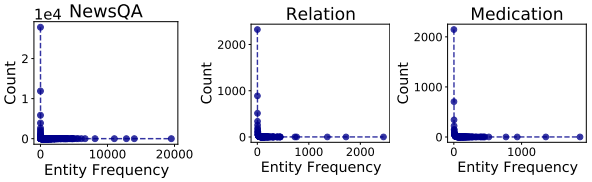


Figure 13: Distribution of frequency of entities on QA datasets: NewsQA, Relation, and Medication, where almost all entities appear less than 10 times, while an extremely few numbers of entities appear very frequently.

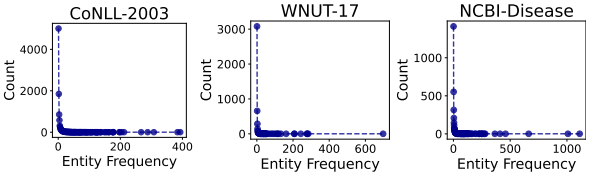


Figure 14: Distribution of frequency of entities on NER datasets: CoNLL-2003, WNUT-17, and NCBI-Disease, where almost all entities appear less than 10 times, while an extremely few numbers of entities appear very frequently.

C.7 Experimental Results on RoBERTa

Although we believe our experimental results on Table 1, 2, and 5 with BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) are enough to show the effectiveness of KALA across different pre-trained language models (PLMs), one might be curious that KALA can work on even other PLMs such as RoBERTa (Liu et al., 2020). Thus, to address such concerns, we additionally conduct experiments on RoBERTa. As shown in Table 12, we observe that our KALA outperforms all baselines except for one case (Fine-Tuning + *more params* on NewsQA). Thus, we believe that our KALA would be useful to any PLMs, not depending on specific PLMs.

D Additional Visualization Results

Here we provide the frequency distribution of entities, additional case studies, and more illustrations of textual examples and embedding spaces.

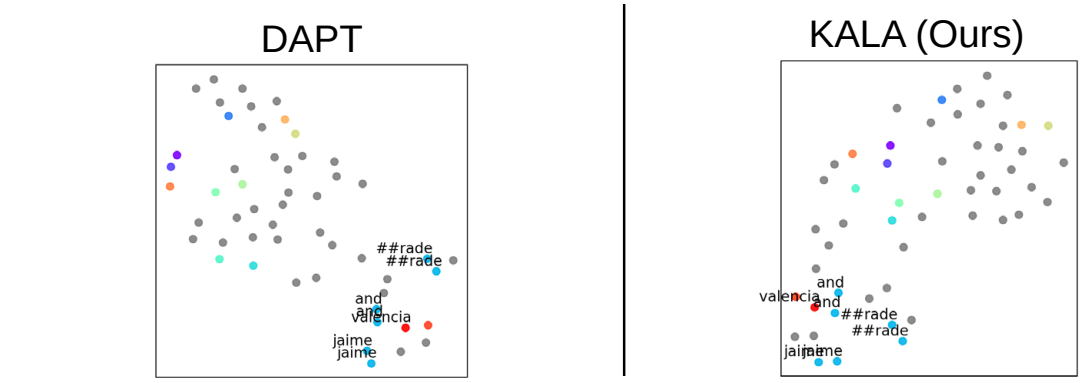
D.1 Additional Representation Visualization

While we already show the contextualized representations of seen and unseen entities in the latent space in Figure 2 right, we further visualize them including the missing baselines of Figure 2, such as Fine-tuning or TAPT, in Figure 12 on the NCBI-Disease dataset. Similar to Figure 2, we observe that all baselines fail to closely embed the unseen entities in the representation space of seen entities. While this visualization result does not give a strong evidence of why our KALA outperforms other baselines, we clearly observe that KALA is beneficial to represent unseen entities in the feature space of seen entities, which suggests that such an advantage of our KALA helps the PLM to generalize over the test dataset, where the context contains unseen entities.

D.2 Entity Frequency Distribution

We visualize the frequency of entities in Figure 13 and 14. The entity frequency denotes the number of mentions of their associated entities within the entire text corpus of the training dataset. As shown in Figure 13 and 14 of QA and NER datasets, the entity frequency follows the long-tail distribution, where most entities appear a few times. For instance, in the NewsQA dataset, more than 20k entities among entire 60k entities appear only once in the training dataset, whereas one entity (*CNN*[10]) appears approximately 20k times. This observation suggests that most of the elements in the entity memory are not utilized frequently. In other words, only few entities are accurately trained with many training instances, whereas there exists the *stale* embeddings which are rarely updated. This observation raises an interesting research question on the efficient usage of the entity memory, as we can see in Figure 8 that the small size of entity memory could result in the better performance (See Appendix C.2). We leave the more in-depth analysis on the entity memory as the future work.

[10] Almost every context in NewsQA includes the text 'CNN' since they are originated from the CNN News.



Context PHOENIX, Arizona (CNN) – Jamie Andrade had just gotten out of the shower when the men came to snatch him. Jamie Andrade was kept in this closet for three days without food or water, police say. His wife, Araceli Valencia, was mopping the kitchen in … (ellipse)…

Facts (*Sampled*)
(Valencia, spouse, Jaime Andrade Jr.)
(Jamie Andrade Jr., spouse, Valencia)

Question who was kidnapped because of her neighbour?
Answer Jaime Andrade

DAPT prediction Araceli Valencia,

KALA prediction Jaime Andrade

Figure 15: A textual example from NewsQA with predictions from each method (DAPT and KALA), and also the T-SNE plot of contextualized representations from the last layer of BERT obtained by each method. Grey dots indicate tokens without any mentions, and dots in other colors indicate tokens with mentions to the entity. We also represent sampled facts in Knowledge Graph we used. Blue text indicates the mention of seen entities and red text indicates the mention of unseen entities. The fact is represented as the format of (head, relation, tail). Text with yellow background indicates the ground truth answer span.

D.3 Additional Case Study

In addition to the case study in Figure 5, we further show the case on the question answering task in Figure 15, like in Section 5.4, With this example, we explain how the factual knowledge in KGs could be utilized to solve the task via our KALA. The question in the example is "who was kidnapped because of **her** neighbor". We observe that DAPT answers this question as *Araceli Valencia*. This prediction may come from matching the word 'her' in the question to the feminine name 'Araceli Valencia' in the context. In contrast, our KALA predicts the *Jaime Andrade* as an answer, which is the ground truth. We suspect that this might be because of the fact "(Jaime Andrade, spouse, Valencia)" in the knowledge graph, which relates the 'Valencia' to the 'Jaime Andrade'. Although it is not clear how it directly affects the model's performance, we can reason that KALA can successfully answer the question by utilizing the existing facts.

D.4 Additional Data Visualization

In Figure 16 and 17, we visualize the examples of the context with its seen and unseen entities and its relational facts. We first confirm that the quality of facts is moderate to use. For instance, in the first example of Figure 16, the fact in the context that *Omar_bin_Laden* is son of *Osama_bin_Laden*, is also appeared in the knowledge graph. In addition, we observe that there are facts that link unseen entities to the seen entities in both Figure 16 and 17. Thus, while some of the facts in the knowledge graph are not accurate, we can represent the unseen entities with their relation to the seen entities. We expect that there is a still room to improve in terms of the quality of KGs, allowing our KALA to modulate the entity representation more accurately. We leave the study on this as the future work.

Context MADRID, Spain (CNN) – One of Osama bin Laden's sons has been denied asylum in Spain, an Interior Ministry spokeswoman told CNN on Wednesday. Omar bin Laden pictured earlier this year during television interview in Rome, Italy. Omar bin Laden, who is in his late 20s, stepped off a plane at Madrid's Barajas International Airport during a stopover late Monday and informed authorities that he planned to request political asylum, the spokeswoman said. Bin Laden has publicly called on his father to abandon terrorism. He prepared his formal asylum request Tuesday at the airport with the help of a translator, filing it around 1 p.m., the spokeswoman said. The Interior Ministry, which had 72 hours to reply to the request, was required to seek the opinion of the U.N. High Commissioner for Refugees on the matter. The UNHCR recommended … (ellipse) …

Facts (*Sampled*)

(Bin Laden, significant event, Flight)

(International Airport, country, Spain)

(International Airport, [UNK], Madrid)

(Omar bin Laden, father, Osama Bin Laden)

(Spain, diplomatic relation, Italy)

(Osama Bin Laden, child, Omar Bin Laden)

(Italy, diplomatic relation, Spain)

Question 1 Where was Omar previously denied?
Answer 1 asylum in Britain.

Question 2 Did Spain give a reason for turning down the asylum?
Answer 2 was given

Question 3 Who was denied asylum in Britain?
Answer 3 Omar bin Laden

Question 4 What family member of Omar bin Laden was associated with terrorism?
Answer 4 his father

Context (CNN) – unseeded Frenchwoman Aravane Rezai produced one of the shocks of the year on Sunday by defeating favorite Venus Williams in straight sets to win the final of the Madrid Open. The 23-year-old Rezai – who had only claimed WTA Tour titles at Strasbourg and Bali prior to Madrid – continued her remarkable week with a 6-2 7-5 victory, adding Williams' scalp to her earlier surprise victories over former world number one's Junstine Henin and Jelena Jankovic. Williams, who returns to No.2 in the world behind younger sister Serena on Monday, lost the opening set in just 27 minutes and then failed to take advantage of a 4-1 lead in the. "I just cannot believe this," world number 24 Rezai – who must now enter calculations for the French Open – told reporters. "Venus played very well and I've always respected her as a player and a champion. I just tried my best today and it worked well for me." Williams, who was looking to secure her 44th career title, only converted two of her 13 break points in the batch – a statistic that contributed greatly to her defeat.

Facts

(Venus Williams, sibling, Aravane Rezai)

(Final, part of, Year)

(Mutua Madrid Open, located in the administrative territorial entity, Madrid)

(Victories, instance of, Military rank)

(Surprise, instance of, Military rank)

(Mutua Madrid Open, instance of, Military rank)

(Final, instance of, Military rank)

Question 1 Which player was the favourite?
Answer 1 Venus Williams

Question 2 Which title number was this?
Answer 2 44th

Question 3 When did the Mardrid Open final take place?
Answer 3 Sunday

Figure 16: NewsQA examples with facts in Knowledge Graph we used in this work. Blue text indicates the mention of seen entities and red text indicates the mention of unseen entities. The fact is represented as the format of (head, relation, tail).

Context The adenomatous polyposis coli (APC) tumour - suppressor protein controls the Wnt signalling pathway by forming a complex with glycogen synthase kinase 3beta (GSK - 3beta), axin / conductin and betacatenin.	**Facts** (*Sampled*) (complex, subclass of, protein) (GSK, instance of, protein) (glycogen, instance of, protein) (APC, instance of, protein)
Context Recently, we reported five Austrian families with generalized atrophic benign epidermolysis bullosa who share the same COL17A1 mutation.	**Facts** (*Sampled*) (epidermolysis bullosa, instance of, mutations)
Context We identified four germline mutations in three breast cancer families and in one breast - ovarian cancer family. among these were one frameshift mutation, one nonsense mutation, one novel splice site mutation, and one missense mutation.	**Facts** (*Sampled*) (frameshift mutation, subclass of, Germline mutations) (Nonsense mutation, subclass of, Germline mutations) (splice site mutation, subclass of, Germline mutations) (missense mutations, subclass of, Germline mutations) (Nonsense mutation, subclass of, cancers) (frameshift mutation, subclass of, cancers) (missense mutations, subclass of, cancers)
Context A nonsense mutation in exon 17 (codon 556) of the RB1 gene was found to be present homozygously in both the retinal and the pineal tumours.	**Facts** (*Sampled*) (retinal, instance of, gene) (Nonsense mutation, subclass of, gene)
Context Sixteen different p16 germline mutations were found in 21 families, while one germline mutation, Arg24His, was detected in the CDK4 gene.	**Facts** (*Sampled*) (p16, subclass of, Germline mutations) (Germline mutations, subclass of, gene) (p16, instance of, gene)
Context Aspartylglucosaminuria (AGU) is a rare disorder of glycoprotein metabolism caused by the deficiency of the lysosomal enzyme aspartylglucosaminidase (AGA).	**Facts** (*Sampled*) (Aspartylglucosaminuria, subclass of, deficiency)
Context Detection of heterozygous carriers of the ataxia - telangiectasia (ATM) gene by G2 phase chromosomal radiosensitivity of peripheral blood lymphocytes.	**Facts** (*Sampled*) (ATM, instance of, gene) (G2 phase, part of, blood) (G2 phase, instance of, gene)
Context HLA typing for HLA - B27, HLA - B60, and HLA - DR1 was performed by polymerase chain reaction with sequence - specific primers, and zygosity was assessed using microsatellite markers.	**Facts** (*Sampled*) (microsatellite, subclass of, primers) (DR1, instance of, microsatellite) (microsatellite, subclass of, typing)

Figure 17: NCBI-Disease examples with facts in Knowledge Graph we used in this work. Blue text indicates the mention of seen entities and red text indicates the mention of unseen entities. The fact is represented as the format of (head, relation, tail).