CapsKG: Enabling Continual Knowledge Integration in Language Models for Automatic Knowledge Graph Completion

Janna Omeliyanenko, Albin Zehe, Andreas Hotho, and Daniel Schlör

Julius-Maximilians-University Würzburg, Am Hubland, 97074 Würzburg, Germany
{omeliyanenko, zehe, hotho, daniel.schloer}@informatik.uni-wuerzburg.de

Abstract. Automated completion of knowledge graphs is a popular topic in the Semantic Web community that aims to automatically and continuously integrate new appearing knowledge into knowledge graphs using artificial intelligence. Recently, approaches that leverage implicit knowledge from language models for this task have shown promising results. However, by fine-tuning language models directly to the domain of knowledge graphs, models forget their original language representation and associated knowledge. An existing solution to address this issue is a trainable adapter, which is integrated into a frozen language model to extract the relevant knowledge without altering the model itself. However, this constrains the generalizability to the specific extraction task and by design requires new and independent adapters to be trained for new knowledge extraction tasks. This effectively prevents the model from benefiting from existing knowledge incorporated in previously trained adapters.
In this paper, we propose to combine the benefits of adapters for knowledge graph completion with the idea of integrating capsules, introduced in the field of continual learning. This allows the continuous integration of knowledge into a joint model by sharing and reusing previously trained capsules. We find that our approach outperforms solutions using traditional adapters, while requiring notably fewer parameters for continuous knowledge integration. Moreover, we show that this architecture benefits significantly from knowledge sharing in low-resource situations, outperforming adapter-based models on the task of link prediction.

Keywords: knowledge graph completion · language model · link prediction · continual learning.

1 Introduction

Our work is at the intersection of two research areas that have received significant attention in recent years: Knowledge Graphs (KGs) and Language Models (LMs). KGs have gained increasing attention over the past decade and have improved the state of the art in natural language approaches through their integration into many downstream tasks such as question answering [31] and sentiment analysis [40]. However, KGs are incomplete and in continuous development. Much

work has been done on the prediction of relations missing between two entities, the so-called link prediction. Moreover, over time, new concepts and relation types emerge that need to be captured in KGs, or even give rise to new KGs. On the other hand, powerful pre-trained LMs such as GPT-2 [24] and BERT [9] have been shown to incorporate a certain amount of factual and common sense knowledge that has been automatically extracted from their unstructured training data, without explicit common sense training. Since this knowledge is extracted from unstructured texts and may not be available in structured form, leveraging these models for KG completion by prompting facts with MASK tokens has emerged in recent research. In the past, pre-trained LMs have been successfully used to improve KGs in two directions: (A) to rate the quality of information contained in KGs [20] and (B) to complete given KGs through fine-tuning these models and conducting link prediction [3,11].

Previous approaches to this have usually fine-tuned the entire language model to all relation types in a KGs at once, which has two main drawbacks: First, fine-tuning the entire LM is very resource intensive, due to the large number of parameters contained in the model. Secondly, if a knowledge graph is updated to include new relation types, or if a new knowledge graph is supposed to be included into the model, the fine-tuning process needs to be redone on the full dataset, rather than only updating it with the new information. For the second case, simply performing fine-tuning on the new relations frequently leads to a problem commonly called *catastrophic forgetting* [17], where the model learns about the new relations, but forgets the ones it had previously learned.

In order to solve these problems, we propose to transfer ideas from the areas of *multi-task learning* and *continual learning*, specifically the use of Adapters and Capsules. Both of these approaches aim to make fine-tuning a model to different sub-tasks (in our case, link prediction for different relation types) as efficient as possible. Adapters (cf. Section 4.2) keep large parts of the model fixed during fine-tuning, allowing to retain the originally learned information within the LM. However, they do not allow for continuously fine-tuning the knowledge contained in a previously trained adapter when new knowledge becomes available or new aspects and sub-tasks become relevant for which information from the LM shall be extracted. The knowledge contained within the adapter to be retrained or fine-tuned to the new dataset would still be subject to catastrophic forgetting. Since KGs are continuously evolving and new facts emerge or new relations become important that can be queried from the LM, new adapters can be trained on these new facts. However, multiple adapters do not communicate with each other even if their purpose is to extract similar information and they could therefore potentially benefit from each other's training progress. Capsules additionally add the capability of sharing knowledge between different sub-tasks, enabling the model to make use of previously learned information (cf. Section 4.2).

We transfer and combine these ideas to the task of link prediction by modelling each relation type in a KG as one sub-task and fine-tuning on them iteratively. This enables both of the desiderata described above: We avoid the high cost of updating the entire language model and we can update a trained

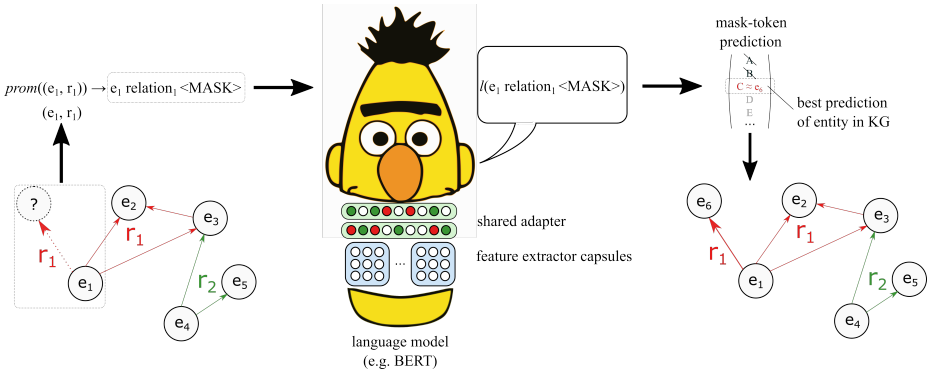

Fig. 1. Overview of our continual relation learning approach within CapsKG. To predict links within a given KG, a query containing a known entity and relation from the KG is obtained and converted to natural language prompt with a <MASK> token. This prompt is forwarded to a relation adapted LM to obtain likely candidate entities. Finally, a filter ensures that predicted candidates correspond to entities contained in the KG. For a detailed view of the relation adapted LM architecture, see Figure 2.

Capsule-based model with new relation types by simply fine-tuning the trained model with the new relations. An overview of the task and our proposed model is depicted in Figure 1.

Our resulting model CapsKG demonstrates the benefit of applying continual learning to the task of KG link prediction on three common KG datasets, WN18, YAGO3-10, and FB15k. Results show consistent improvement across all datasets, comparable to existing related work [36], while requiring significantly fewer trainable parameters. CapsKG also shows superior results in a low resource setting, when only small amounts of data are available for training, especially when relation types are similar in the sense of sharing similar entities or relations are semantically similar. [1]

The remainder of this work is structured as follows: Section 2 introduces the scientific context of link prediction through LMs, while Section 4 describes the chosen adapter-based capsule network in detail. Experiments on multiple datasets are conducted and discussed in Section 5. Section 6 concludes the paper.

2 Related Work

The task of completing KGs via link prediction is a well-established field of research within the semantic web community that aims to explore new knowledge by extrapolating from observed existing facts [7]. This task is commonly approached through learning KG representations, e.g. through classical translation-based or tensor factorization-based graph embedding techniques, such as TransE

[1] Our code is available at https://professor-x.de/code-capskg.

[33] or RotatE [39], as well as complex neural network architectures [7]. In this context, capsule networks have been used as a specialized architecture that extracts low level features to improve graph embeddings [19,32]. However, their usage in previous work is so far limited to capsules as simple feature extractors. Additionally, within traditional KG representation learning, several works have identified the use of so called continual learning schemes, where architectures are designed to be continuously trained on newly incoming training data without forgetting previously learned information [8,26].

Beyond the use of traditional KG embedding techniques, a recent trend of works has identified the potential of large LMs for link prediction in KGs. In the past, LMs have been effectively utilized for two purposes: testing the quality of existing information contained in KGs [20], and extracting additional facts to complete KGs [11]. Successful applications of LMs for KG link prediction convert factual triples from a KG to natural language either through relation-specific sentence templates or through leveraging available textual descriptions of entities [3,5,11,12,22,27,30,35,37,38]. The resulting textual inputs are then used to fine-tune pre-trained LMs to extract predictions of likely entities.

Wang et al. [31] have further improved this procedure by incorporating adapters, which is an architecture designed to prevent catastrophic forgetting in LMs. In this architecture, additional neuron layers are inserted between the layers of a pre-trained LM for fine-tuning, while all weights belonging to the original LM are frozen to prevent overwriting learned information. This approach allows the LM to retain its learned high quality language representations, but does not allow for the training of multiple adapters that can exchange information and thus support each other.

Pfeiffer et al. [23] address this issue by proposing AdapterFusion, a two stage training process that first trains multiple adapters and afterwards learns a knowledge composition through training data. While this approach is capable of aggregating the knowledge of multiple adapters, it does not allow individual adapters to support each other during training. Moreover, it is not suitable for continual learning, as the entire composition training process has to be repeated when new training data is available.

In contrast to these existing works, our work jointly leverages continual learning strategies, large LMs, and adapters for KG link prediction. This allows for the use of powerful LMs for KG link prediction, while continuously integrating new information into the learned model and facilitating communication of individual adapters during the training process.

3 Task Definition: Link Prediction

KGs are structural representations of real-world factual knowledge in the form of triples (e_1, r, e_2) where e_1 and e_2 represent the head and tail entities describing world objects and r represents a specific relation connecting these two entities, for example (`Berlin`, `CapitalOf`, `Germany`). Although existing KGs such as YAGO and WikiData cover a large portion of factual knowledge, they often suffer from

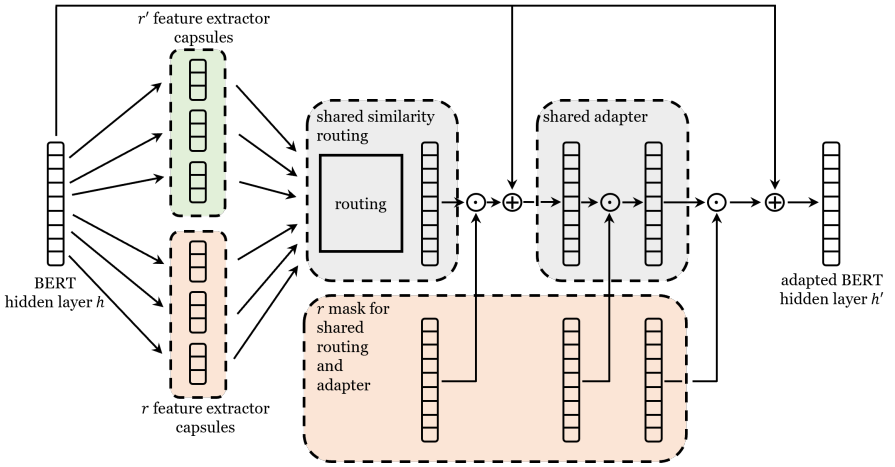

Fig. 2. Continual relation learning adapter showcasing two exemplary relations. The adapter is attached to LM hidden layer h, where relation-specific features are extracted. Feature extractors of a previously trained relation r' (green) may influence the currently trained relation r (orange) through similarity routing. Results are passed through a shared adapter (grey) that is biased through relation-specific masks. Note that most weights in the architecture are contained within the shared layers, leading it to share most trainable weights.

incompleteness [10,34]. To address this problem, link prediction was introduced that aims to learn a function f that predicts missing relations between entities in a given graph. Given a head entity e_1 and a relation r, the function f predicts the tail entity e_2 that is most likely to be connected via this relation:

$$f_{KG}(e_1, r) \rightarrow e_2 \tag{1}$$

4 Methodology

In this chapter, we describe in detail the architecture we have chosen to solve the link prediction task by continuously adapting an LM to the different knowledge sub-tasks. The architecture is also illustrated in Figure 2.

4.1 Base Model: LM for Link Prediction

For our base model, we follow the idea of prompting a pre-trained LM for the task of link prediction [3,22]. To access the model's knowledge, facts are converted into a natural language representation in the form of a sequence of tokens, where the desired unit of information is omitted or masked and the model is prompted to complete the given text sequence. For example, given a tuple (`Berlin`, `CapitalOf`), we transform the tuple into the following (masked) sentence: "Berlin is a capital of <MASK>." This sentence is then input to the LM,

which predicts the most likely token for <MASK> – in this case `Germany`. Note that the quality of the extracted information strongly depends on the prompt sentence, and therefore the generation of well-performing prompts has been focus of several scientific works [4,13,25]. More formally, given a function *prom* that maps tuples (e_1, r) to a natural language representation prompt p, and a pre-trained LM l, prompting for link prediction is defined as follows:

$$prom((e_1, r)) \rightarrow p, \qquad l(p) \rightarrow e_2 \tag{2}$$

In the example above, this becomes $prom((\texttt{Berlin}, \texttt{CapitalOf})) \rightarrow$ "Berlin is a capital of <MASK>." $=: p$, and $l(p) \rightarrow$ `Germany`.

We fine-tune this model to the task of link prediction by converting all tuples in the train set of a dataset (cf. Section 5.1) to natural language sentences and performing regular masked LM training. However, this approach comes with the disadvantage that the model is prone to catastrophic forgetting, thereby potentially losing its generalizability and language modeling performance, as well as missing the possibility of extending the trained model with additional relation types.

4.2 Continual Learning for Link Prediction

Continual learning [21] studies this problem of avoiding catastrophic forgetting, i.e., the general ability of the system to maintain its performance in a previously learned task or domain without access to the previous training data while learning a new task. This becomes relevant to incorporating LMs for KG completion from two perspectives. First, the knowledge that the LM has learned during its pre-training and training to perform natural language modeling should be retained and not forgotten, although the specific domain of the KG including the different relations, relevant entities and grammar cover only a very small subset of the previously acquired knowledge. Second, the KGs themselves continuously expand. This means that by the integration of new entities or relations, the previously learned KG knowledge might also become subject to catastrophic forgetting.

One solution to address both issues following the idea of continual learning is the adapter architecture [31]. This architecture has been proven useful as it allows to freeze the underlying LM while training new knowledge into specific adapter layers, added to the original architecture. Furthermore, new adapters can be added for new relations and facts to be learned without altering the weights of other adapters.

Basic Adapter To be precise, the adapter [31] is a small set of additional trainable neural layers that are inserted between the layers of a pre-trained base LM l to learn new features relevant for a new task, for example, link prediction for a new relation type r. During training, the base model is frozen and only the parameters of the adapter are trainable. In our work, following [15], the adapter consists of two fully connected layers, which are inserted into each transformer

layer in our LM. The input to the adapter is the output of the intermediate layer $h^{(r)} \in \mathbb{R}^{d_s \times d_e}$ of the base model, with the sequence length d_s and the dimension of the base model's hidden state output d_e. Adapter layers map the input to an intermediate adapter dimensionality free of choice and back to the hidden state space of the model. Finally, the intermediate output of the original base model is added via skip connections, where f_c and f_d are fully connected layers by

$$h'^{(r)} = f_c(f_d(h^{(r)})) + h^{(r)}. \tag{3}$$

Shared Adapter In a basic adapter architecture, a new adapter is added for each type of relation. With a growing number of relations, this highly increases the number of trainable parameters, as for each relation a separate adapter with independent weights is provided. Inspired by [15], we propose to share the adapter layers for all relation types making the model more parameter efficient with increasing number of relations. To prevent catastrophic forgetting that may occur when continuously training adapter layers with new data, a masking procedure is introduced into the adapter. For each relation r, a mask $m^{(r)}$ with same dimensionality as each adapter layer is calculated from relation embeddings $e^{(r)}$ based on the task's id. These relation embeddings are then converted to a pseudo-gating function through a Sigmoid activation and a scale hyper-parameter s as follows:

$$m^{(r)} = \sigma(se^{(r)}) \tag{4}$$

The hyper-parameter s is chosen as positive scalar that gradually increases in value to $\gg 1$ during training, forcing the learned mask $m^{(r)}$ to contain values closer to 0 or 1. The resulting pseudo-binary masks $m_d^{(r)}$ and $m_c^{(r)}$ for the adapter layers f_c and f_d are multiplied element-wise with the respective shared adapter layer, giving:

$$h'^{(r)} = f_c(f_d(h^{(r)}) \otimes m_d^{(r)}) \otimes m_c^{(r)} + h^{(r)} \tag{5}$$

As we observe that the pseudo-gate function commonly produces non binary masks that still lead to catastrophic forgetting, we additionally follow [14] and binarize all masks for a sub-task once this sub-task has been fully trained.

$$m_{eval}^{(r)} = \begin{cases} 1 & \text{if } \sigma(se^{(r)}) > 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

To prevent the gradient flow through neurons that were already used by previously learned tasks, or in this case relation types, the overall use of neurons by previous task masks is calculated by

$$m^{(prev)} = \text{MaxPool}(\{m_{eval}^{(r')},\ r' \in \{\text{previously trained } r\}\}) \tag{7}$$

and applied to the gradients $g^{(r)}$ to limit the updates on these neurons during backpropagation through

$$g'^{(r)} = g^{(r)} \otimes (1 - m^{(prev)}). \tag{8}$$

As we calculate previous masks $m^{(prev)}$ for gradient gating from these binarized masks, this allows trained relations to truly reserve neurons within the adapter preventing any gradient updates that could cause forgetting. Conversely, other relations may still utilize reserved neurons during the forward pass, allowing them to use shared information of previously learned relations while preventing problematic updates.

As trained relations are capable of reserving neurons through the masks which prevents them from being trained by new relations, sparse masks are required to prevent a single relation from occupying the entire architecture. To achieve sparse masks, a regularization term is added to the loss function $\mathcal{L}$, which regularizes the values of $m^{(r)}$ during training depending on the number of neurons that are not already occupied by previously trained relations:

$$\mathcal{L}' = \mathcal{L} + \lambda \cdot \frac{\|m^{(r)}\|_1}{d_m - \|m^{(r)}_{eval}\|_1}, \tag{9}$$

where d_m denotes the dimensionality of $m^{(r)}$ and λ is a weighting hyperparameter.

To summarize, this shared adapter architecture allows parameter efficient continual learning and the utilization of weights from previously learned relations if they are helpful without risking catastrophic forgetting as these weights are not changed during back-propagation. This idea is further extended by capsules as low-level feature extractors, which instead of masking rely on a routing mechanism. This approach brings the flexibility to incorporate additional capsules in a trained model when required for continual learning.

Relation Specific Capsules with Knowledge Transfer To enable further information transfer and reuse of previously learned knowledge between relation types, we apply a capsule architecture to train low-level feature extractors with a similarity based routing to share extractors between relation types [15]. Relation specific capsules consist of three components: relation capsule layer, transfer capsule layer and a transfer routing mechanism. Relation-specific extractor capsules f_i in the relation capsule layer are used as low-level feature extractors that learn relation-relevant information from intermediate layers of a given base model through

$$p_i^{(r)} = f_i(h^{(r)}), \tag{10}$$

parameterized as small fully connected layers. Capsules in the transfer capsule layer are used to represent transferable features extracted by relation specific capsules. A special routing mechanism consisting of several steps is used to transfer features between lower layer relation capsules and higher layer transfer capsules. To share the extracted features between relation capsule $p_i^{(r)}$ and a transfer capsule $t_j^{(r)}$, a pre-routing vector is generated that extracts relevant features from relation capsule i for transfer capsule j as

$$u_{j|i}^{(r)} = W_{ij} p_i^{(r)}, \tag{11}$$

where $W_{ij} \in \mathbb{R}^{d_c \times d_k}$ is a transformation matrix learned during training, with d_c denoting the dimension of relation capsule $p_i^{(r)}$ and d_k denoting the dimension of transfer capsule $t_j^{(r)}$.

In order to transfer useful knowledge between relation types, the similarity between relation types i and the relation type r is estimated using multiple convolution layers that form a learned similarity estimator comparing the relation capsules of i and r:

$$q_{j|r}^{(r)} = \text{MaxPool}(\text{ReLU}(u_{j|r}^{(r)} * W_q + b_q)) \tag{12}$$

$$a_{j|i}^{(r)} = \text{MaxPool}(\text{ReLU}(u_{j|i}^{(r)} * W_a + f_a(q_{j|r}^{(r)}) + b_a)) \tag{13}$$

Convolutions over $u_{j|r}$ and $u_{j|i}$ extract the relevant features of the relation type r and i respectively. The similarity score $a_{j|i}^{(r)}$ is calculated in Equation (13), where f_a is a linear layer for matching the dimensions of $q_{j|r}^{(r)}$ and $u_{j|i}^{(r)}$, $b_a, b_q \in \mathbb{R}$ are biases, $W_a, W_q \in \mathbb{R}^{d_e \times d_\omega}$ convolution filters and d_ω denotes the window size.

To keep knowledge exchange constrained to only similar relation types, a binary differentiable gating function is calculated using convolution with Gumbel Softmax

$$\delta_{j|r}^{(r)} = \text{Gumbel\_softmax}(a_{j|i}^{(r)} * W_\delta + b_\delta), \tag{14}$$

resulting in a gating function for similar tasks. The overall output is obtained by aggregating over the element-wise product of all similar extractor capsules with their corresponding learned similarities to the relation that is currently trained by

$$v_j^{(r)} = \sum_{i=1|\delta_{i|j}^{(r)}=1}^{n+1} v_{j|i}^{\text{tran}(r)}, \qquad v_{j|i}^{\text{tran}(r)} = a_{j|i}^{(r)} \otimes u_{j|i}^{(r)} \tag{15}$$

with $\delta_{j|r}^{(r)} \in \{0\text{: disconnected, } 1\text{: connected}\}$. Note that this gating allows for gradient updates to apply to the extractor capsules of previously learned relation types if similarity to the current relation type is found by the Gumbel Softmax, thus facilitating a backward knowledge transfer to previously trained relation types. We additionally investigate model performance when preventing these updates even on similar relations following [15], allowing only forward knowledge transfer from previously learned relations to the new relations.

The generated output $v^{(r)}$ is forwarded through a fully connected layer f_b to match the base model's hidden dimension. The base model's intermediate output $h^{(r)}$ is added again through a skip connection, giving

$$v'^{(r)} = f_b(v^{(r)}) + h^{(r)}. \tag{16}$$

Finally, $v'^{(r)}$ contains the aggregated relation-specific information of all extractor capsules and is given as input to the shared adapter.

Table 1. Dataset characteristics of the used link prediction datasets after filtering according to Section 5.1.

dataset	relation types	entities	train triples	dev triples	test triples	total
WN18	5	23324	27045	9017	9018	45080
YAGO	8	92991	75774	25260	25262	126296
FB15k	18	10754	38178	12734	12736	63648

5 Experiments

In the following, we showcase the introduced architecture on three KG link prediction datasets.

5.1 Datasets

For our experiments, we use three established KG datasets commonly used in link prediction settings, namely (1) the WN18 dataset [2] with triples from the WordNet KG, (2) the YAGO3-10 dataset [18,29] (called YAGO from hereon) with triples from the YAGO KG, and (3) the FB15k dataset [28] that contains data from the FreeBase KG. WN18 has a strict hierarchical structure, with most triples representing the hyponym and hypernym relation. YAGO is a subset of YAGO3, with most triples covering descriptive factual information about people, including birthplace, nationality, and more. FB15k is a subset of FreeBase that contain a rich collection of factual information about a diverse range of subjects including movies, actors, people, places and more.

Following previous work that prompts LMs through the use of entity masking, we remove all triples from the datasets where the target entity is (1) larger than one token or (2) not known within the LM's vocabulary [6,11,22]. As we evaluate the distinct relations within the dataset separately, which results in very small and unbalanced splits for some relations after filtering, we join the original train, development, and test splits to one dataset, remove all relations that contain less than 1200 triples, and split the remaining relations back into train, development, and test sets of 60%, 20%, and 20% respectively. The statistics of the resulting datasets can be found in Table 1.

5.2 Experimental Setup

In this section, we describe our evaluation setup, including the sentence generation, evaluation metrics, methods that we evaluated, as well as hyperparameters used.

Sentence Generation To convert the KG triples into natural language (*prom* in Equation (2)), we use the sentence templates of [22] where applicable and design sentence templates in the same style for relations not covered by their work. The full list of used templates is listed in the repository.

Evaluation Metric To evaluate link prediction performance of the investigated models, we use the established hits at one accuracy (Hits@1), which corresponds to the ratio of exact matches between prediction and correct KG entity. Since there may be multiple correct completions e_2 for a given tuple (e_1, r), we follow [11] and filter out all correct alternative entities from the model prediction to avoid penalizing the models for the correct answers that differ from the current label entity. More formally, let $P = l(prom(e_1, r))$ be the probability distribution over tokens returned by the LM l. We count the prediction as correct if

$$e_2 = argmax_e \{P(e) \mid (e_1, r, e) \notin (\text{train} \cup \text{test}) \backslash \{(e_1, r, e_2)\}\}. \tag{17}$$

Scores are always reported as averages over all sub-tasks. To prevent results being skewed by statistical fluctuation, we repeat all experiments with 5 random seeds and report means and standard deviations of all metrics.

Models While the general adapter framework is LM agnostic, we use a pre-trained BERT model in this experimental setup due to its public availability, computational efficiency, and established use within the domain of KG completion [5,11]. We compare the following models in our experiments:

BERT$_{\text{frozen}}$ A BERT base model that is prompted without any fine tuning.
BERT An independent BERT base model is trained and evaluated for each relation type [11].
BERT-CL A BERT base model is iteratively trained for all relation types and evaluated once at the end.
Adapter An independent BERT base model with frozen parameters and one trainable Adapter is trained and evaluated for each relation type [9,31].
Adapter-CL A frozen BERT base model with one trainable Adapter is iteratively trained for all relation types and evaluated once at the end.

Table 2. Number of trainable model parameters for the the YAGO, WN18, and FB15k data. The number of parameters depends on the number of relation types in the dataset, since in the non-CL settings, a separate model is trained for each relation type, while in the CL settings, additional Capsules are added to the model for each relation type.

Dataset	Model	specific per r (M)	shared across r (M)	total (M)
WN18	BERT	301.99	0.00	1,509.95
	Adapter	73.83	0.00	369.16
	CapsKG	0.16	75.53	76.33
YAGO	BERT	301.99	0.00	2,415.92
	Adapter	73.83	0.00	590.65
	CapsKG	0.16	75.53	76.81
FB15k	BERT	301.99	0.00	5,435.83
	Adapter	73.83	0.00	1,328.96
	CapsKG	0.16	75.53	78.41

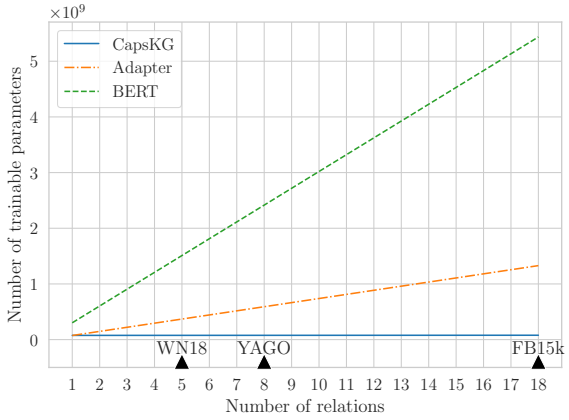

Fig. 3. Visualization showing trainable parameters in billions for BERT, Adapter and CapsKG on an increasing numbers of relations, marking the datasets used in this study. CapsKG scales very conservatively due to the large numbers of shared parameters.

CapsKG$_{\text{forward}}$ The full CapsKG model (Section 4.2) is iteratively trained for all relation types and evaluated once at the end, *not* allowing the capsules for previous relation types to be updated even if similarity to the current relation type is given by the Gumbel Softmax.

CapsKG$_{\text{backward}}$ The full CapsKG model (Section 4.2) is iteratively trained for all relation types and evaluated once at the end, allowing the capsules for previous similar relation types to be updated.

In BERT and BERT-CL, the entire model is updated during fine-tuning, leading to a very expensive training process and very high capacity of the model. In Adapter and Adapter-CL, the number of trained parameters is greatly reduced, leading to a more efficient training process. The resulting number of trainable parameters of used models on all datasets is reported in Table 2. We additionally visualize the increase in trainable parameters for each model with increasing numbers of relations in Figure 3. The CapsKG-variations use Capsules to enable the model to share knowledge between the different relation types and simultaneously avoid catastrophic forgetting. Note that CapsKG still requires increasing numbers of trainable parameters when adding new relations, which is required to prevent catastrophic forgetting [31]. Nevertheless, the increase in trainable parameters per relation is vastly reduced in CapsKG in comparison to other models, due to the efficient parameter sharing.

Model Hyperparameters For the adapter and capsule models, we follow the setup of [15]. We use an adapter size of 2000, adding adapter modules before each layer normalization within the 12 BERT transformer layers. Knowledge extractor capsules are set to 3 extractor capsules of hidden size 3 per relation. Training was carried out on a single A100 GPU for 30 epochs using a regularization

parameter λ (cf. Equation (9)) of 10, a batch size of 64, and learning rate of 5e-5 using the Adam optimizer [16], choosing the model with lowest validation loss after training each relation.

5.3 Link Prediction Evaluation

We evaluate our proposed CapsKG architecture on a link prediction task for three different KG datasets, WN18, YAGO, and FB15k to systematically investigate its benefit compared to previous architectures.

The results of this comparison on all three datasets are shown in Table 3, measured through Hits@1 accuracy on the test set.

The first section with $\text{BERT}_{\text{frozen}}$ shows that prompting a pre-trained LM without any training on the KG data results in poor performance across all datasets. Due to the significantly lower performance of $\text{BERT}_{\text{frozen}}$ compared to all other approaches, we omit this baseline in the following experiments.

The second section comprises BERT and Adapter and represents the resource intensive training of one individual model per relation. As these models are trained independently by design, they denote the performance achievable without leveraging inter-relation dependencies and are evaluated without the threat of catastrophic forgetting. In comparison to BERT, the Adapter-based model performs slightly better on all datasets, despite its significantly lower number of trainable parameters (cf. Table 2). This highlights the potential of the general model decision to freeze the underlying LM and incorporate additional trainable layers, even if no continual learning setting with the danger of catastrophic forgetting is given.

The third block of our results contains all models trained in a continual learning setting, including both variants of our CapsKG. By iteratively incorporating new relations and presenting the respective training data to the model, this setting is prone to catastrophic forgetting. For both of the architectures not specifically optimized for continual learning, training in the continual learning setting as BERT-CL and Adapter-CL shows the severe impact of catastrophic

Table 3. Results for link prediction on WN18, YAGO, and FB15k with 5 random seeds, reporting mean and standard deviation of Hits@1 performance.

Model/Dataset	WN18	YAGO	FB15k
$\text{BERT}_{\text{frozen}}$	10.7 ± 0.0	27.0 ± 0.0	5.5 ± 0.0
BERT	25.8 ± 0.5	48.0 ± 0.1	34.7 ± 0.7
Adapter	26.4 ± 0.5	48.7 ± 0.4	35.6 ± 0.2
BERT-CL	20.2 ± 1.2	40.4 ± 4.0	11.4 ± 3.2
Adapter-CL	18.4 ± 1.2	41.5 ± 0.7	16.3 ± 1.4
$\text{CapsKG}_{\text{forward}}$	$\mathbf{27.4 \pm 0.4}$	49.4 ± 0.3	$\mathbf{36.1 \pm 0.3}$
$\text{CapsKG}_{\text{backward}}$	27.2 ± 0.4	$\mathbf{49.5 \pm 0.3}$	34.9 ± 0.3

forgetting as their performance is impaired significantly for all datasets, occasionally losing more than half of their prediction power. An in-depth analysis of the training course shows that they do indeed suffer from catastrophic forgetting, as the relation most recently learned shows good performance, while the performance of previously learned relations deteriorates over time. The FB15k dataset especially suffers from this forgetting effect, which may be attributed to its large number of different relation types that all have the potential to disrupt trained knowledge of previous relation types. BERT-CL achieves a very low performance on this dataset due to only retaining good prediction accuracy on the relation type trained last, entirely failing to predict previous relation types. Adapter-CL manages to retain more information compared to the BERT-CL model on the FB15k dataset, but still performs significantly worse in this continual learning scenario.

Our CapsKG model, however, performs considerably better in the continual learning setting, strongly outperforming the other approaches throughout all datasets. Additionally, our CapsKG model even consistently outperforms the non-continual models that train on individual relations. This can be attributed to the internal structures that route and reuse previously learned low-level feature extractors. The performance gain, even in comparison to the much more parameter-intensive BERT and Adapter models (cf. Table 2), suggests that iterative training emphasizes the sharing of knowledge from different but related relations reminiscent of the concept of curriculum learning [1].

Finally, we compare both variants of CapsKG, $\text{CapsKG}_{\text{forward}}$ that allows only to use parameters of previously learned relation types and $\text{CapsKG}_{\text{backward}}$ that allows to update the model parameters of knowledge capsules from previously trained relation types. One can observe that both models perform comparably with $\text{CapsKG}_{\text{forward}}$ slightly outperforming $\text{CapsKG}_{\text{backward}}$ which might be a consequence of the significantly larger number of relation types in FB15k accumulating disturbance on early learned relations over the course of the training.

Overall our results highlight the superior performance of the CapsKG architecture on all link prediction datasets.

5.4 Low Resource Evaluation

Since [15] found the capsule architecture to be more efficient in low-resource scenarios where shared knowledge may be used to overcome the limited training data, we further evaluate the model's performance with only a limited number of training instances for each relation. This is particularly relevant for the task of KG-completion, as some of the knowledge graph (KG) relations are very sparse.

To evaluate the performance of our model in a low-resource setting, we build low-resource subsets of the datasets used in the previous experiment by reducing the number of training samples to only 200 triples per relation. We follow the architectural parameters and experimental setup of our previous experiments, other than that we set the regularization parameter λ to a large number of 2000, as we observe that the limited number of training samples severely impedes the

Table 4. Link prediction on WN18, YAGO, and FB15k using only 200 triples for training each relation, repeated with 5 seeds and reporting mean and standard deviation of Hits@1 performance.

Model/Dataset	$WN18_{200}$	$YAGO_{200}$	$FB15k_{200}$
BERT	19.1 ± 0.5	45.8 ± 0.6	30.7 ± 0.5
Adapter	18.8 ± 0.2	44.9 ± 0.7	32.2 ± 0.6
BERT-CL	13.1 ± 0.5	29.5 ± 6.5	5.5 ± 0.0
Adapter-CL	13.6 ± 0.4	26.1 ± 1.1	6.7 ± 1.1
$CapsKG_{forward}$	$\mathbf{19.7 \pm 0.4}$	$\mathbf{46.4 \pm 0.2}$	$\mathbf{33.7 \pm 0.1}$
$CapsKG_{backward}$	19.1 ± 0.4	46.3 ± 0.2	32.2 ± 0.2

regularization of weight sharing (cf. Equation (9)) within CapsKG. We train the model on a single RTX 2080 ti using a batch size of 16.

The results of this low resource evaluation are summarized in Table 4. CapsKG outperforms BERT and Adapter on all datasets, while requiring fewer parameters. Note that both architectures have been explicitly fine-tuned for each relation individually. This suggests that the parameter sharing and routing between similar feature extractor capsules allows the model to access knowledge extracted by previously trained relations that a model trained individually per relation cannot leverage.

The remarkably lower performances for BERT-CL and Adapter-CL suggest, that even though the number of training samples has been reduced by magnitudes, the effect of catastrophic forgetting is not mitigated in this setting, such that this effect outweighs the lower number of training samples overall.

5.5 Topic Evaluation

To verify our hypothesis that relation types learned with CapsKG iteratively profit from previously learned similar relations, we conduct a topic evaluation experiment in the low-resource setting from Section 5.4. We therefore sample four topic groups from the FreeBase hierarchy, *people*, *music*, *sport*, and *film*, each consisting of more than one relation type, topically related. The goal is to examine to what extent CapsKG based models benefit from topically similar relations while *similar* can be understood in terms of overlap of similar entity pairs and semantics from the LM. The results of the experiments are shown in Table 5 and support our hypothesis, as CapsKG outperforms all other models. The largest improvement is observed in the smallest topic group consisting of only two relation types which are the direct inverse relations and thus highly relevant for each other: location/contains and location/containedby. Our experiment shows that, when considering topic-specific similar relations, our model can leverage knowledge from prior relations for all topic groups, showing its potential in the low-resource continual learning setting.

Table 5. Link prediction on FB15k relations grouped by topic according to the KG relation types, each with 5 random seeds.

Model/Topic	people	music	location	film
BERT	41.1 ± 1.6	20.7 ± 1.3	48.0 ± 1.2	36.4 ± 1.6
Adapter	42.4 ± 0.7	23.4 ± 0.7	47.4 ± 0.4	37.9 ± 0.6
CapsKG-Forward	43.8 ± 0.4	$\mathbf{24.2 \pm 0.3}$	50.8 ± 0.4	$\mathbf{38.8 \pm 0.5}$
CapsKG-Backward	$\mathbf{44.0 \pm 0.3}$	22.9 ± 0.8	$\mathbf{50.9 \pm 0.4}$	38.7 ± 0.2

6 Conclusion

In this work, we have proposed to transfer architectures from continual learning to KG completion using LMs, and demonstrated the performance of an Adapter- and Capsule-based architecture on the task of link prediction in KGs in CapsKG. CapsKG combines the benefits of adapters for the extraction of knowledge from LMs with capsules as feature extractors to perform link prediction in KGs in a continual learning scenario. Our experiments on three common KG datasets, WN18, YAGO, and FB15k demonstrate the benefit of the proposed architecture, outperforming even fine-tuned BERT that has 20 to 70 times more trainable parameters and Adapters that have 5 to 17 times more trainable parameters. We have also shown that our CapsKG model maintains and partially even improves in performance when trained in a continual learning scenario, while other models show the devastating behavior of catastrophic forgetting. Our low-resource experiment demonstrates the benefit of knowledge sharing across different sub-tasks, especially when only few training instances per sub-task are available.

Our findings suggest that our proposed model-agnostic architecture allows continual learning for KG completion using LMs even in low resource scenarios, making it a promising foundation for future research in the Semantic Web community. In our consecutive research, we plan to systematically evaluate a variety of base-LMs with CapsKG on top, to examine their potential for generic KG completion as well as link prediction with specific domain adapted LMs in their particular domain, for example for medical KGs. Moreover, different approaches to precisely model similarity for the low-level feature extraction capsules and their routing mechanism is an interesting field for future work as well.

Supplemental Material Statement: Source code for CapsKG and used baselines is available at https://professor-x.de/code-capskg. The used datasets are publicly available at https://everest.hds.utc.fr/doku.php?id=en:transe for both WN18 and FB15k, and at https://pykeen.readthedocs.io/ for YAGO3-10.

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning. pp. 41–48 (2009)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: Advances in Neural Information Processing Systems. vol. 26. Curran Associates, Inc. (2013)
3. Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., Choi, Y.: COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4762–4779. Association for Computational Linguistics, Florence, Italy (Jul 2019)
4. Bouraoui, Z., Camacho-Collados, J., Schockaert, S.: Inducing relational knowledge from bert. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7456–7463 (2020)
5. Brayne, A., Wiatrak, M., Corneil, D.: On Masked Language Models for Contextual Link Prediction. In: Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures. pp. 87–99. Association for Computational Linguistics, Dublin, Ireland and Online (May 2022)
6. Cao, B., Lin, H., Han, X., Sun, L., Yan, L., Liao, M., Xue, T., Xu, J.: Knowledgeable or Educated Guess? Revisiting Language Models as Knowledge Bases (Jun 2021)
7. Dai, Y., Wang, S., Xiong, N.N., Guo, W.: A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. Electronics **9**(5), 750 (May 2020)
8. Daruna, A., Gupta, M., Sridharan, M., Chernova, S.: Continual Learning of Knowledge Graph Embeddings. IEEE Robotics and Automation Letters **6**(2), 1128–1135 (Apr 2021)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (May 2019)
10. Ebisu, T., Ichise, R.: Toruse: Knowledge graph embedding on a lie group. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
11. Fichtel, L., Kalo, J.C., Balke, W.T.: Prompt Tuning or Fine-Tuning - Investigating Relational Knowledge in Pre-Trained Language Models. In: 3rd Conference on Automated Knowledge Base Construction (Sep 2021)
12. Hao, S., Tan, B., Tang, K., Ni, B., Zhang, H., Xing, E.P., Hu, Z.: BertNet: Harvesting Knowledge Graphs from Pretrained Language Models (Dec 2022)
13. Haviv, A., Berant, J., Globerson, A.: Bertese: Learning to speak to bert. arXiv preprint arXiv:2103.05327 (2021)
14. Ke, Z., Lin, H., Shao, Y., Xu, H., Shu, L., Liu, B.: Continual training of language models for few-shot learning. arXiv preprint arXiv:2210.05549 (2022)
15. Ke, Z., Liu, B., Ma, N., Xu, H., Shu, L.: Achieving Forgetting Prevention and Knowledge Transfer in Continual Learning. In: Advances in Neural Information Processing Systems. vol. 34, pp. 22443–22456. Curran Associates, Inc. (2021)
16. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (Dec 2014)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)

18. Mahdisoltani, F., Biega, J., Suchanek, F.M.: YAGO3: A Knowledge Base from Multilingual Wikipedias
19. Nguyen, D.Q., Vu, T., Nguyen, T.D., Nguyen, D.Q., Phung, D.: A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 2180–2189. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
20. Omeliyanenko, J., Zehe, A., Hettinger, L., Hotho, A.: LM4KG: Improving Common Sense Knowledge Graphs with Language Models. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) The Semantic Web – ISWC 2020. pp. 456–473. Lecture Notes in Computer Science, Springer International Publishing, Cham (2020)
21. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural networks **113**, 54–71 (2019)
22. Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.H., Riedel, S.: Language Models as Knowledge Bases? (Sep 2019)
23. Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I.: AdapterFusion: Non-Destructive Task Composition for Transfer Learning (Jan 2021)
24. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language Models are Unsupervised Multitask Learners (2019)
25. Shin, T., Razeghi, Y., Logan IV, R.L., Wallace, E., Singh, S.: Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980 (2020)
26. Song, H.J., Park, S.B.: Enriching Translation-Based Knowledge Graph Embeddings Through Continual Learning. IEEE Access **6**, 60489–60497 (2018)
27. Teru, K.K., Denis, E., Hamilton, W.L.: Inductive Relation Prediction by Subgraph Reasoning (Feb 2020)
28. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality. pp. 57–66. Association for Computational Linguistics, Beijing, China (Jul 2015)
29. Tran, H.N., Takasu, A.: MEIM: Multi-partition Embedding Interaction Beyond Block Term Format for Efficient and Expressive Link Prediction (Oct 2022)
30. Wang, B., Shen, T., Long, G., Zhou, T., Wang, Y., Chang, Y.: Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In: Proceedings of the Web Conference 2021. pp. 1737–1748. ACM, Ljubljana Slovenia (Apr 2021)
31. Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., ji, J., Cao, G., Jiang, D., Zhou, M.: K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters (Dec 2020)
32. Wang, Y., Xiao, W., Tan, Z., Zhao, X.: Caps-OWKG: A capsule network model for open-world knowledge graph. International Journal of Machine Learning and Cybernetics **12**(6), 1627–1637 (Jun 2021)
33. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge Graph Embedding by Translating on Hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence **28**(1) (Jun 2014)
34. West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., Lin, D.: Knowledge base completion via search-based question answering. In: Proceedings of the 23rd international conference on World wide web. pp. 515–526 (2014)

35. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for Knowledge Graph Completion (Sep 2019)
36. Youn, J., Tagkopoulos, I.: Kglm: Integrating knowledge graph structure in language models for link prediction. arXiv preprint arXiv:2211.02744 (2022)
37. Youn, J., Tagkopoulos, I.: KGLM: Integrating Knowledge Graph Structure in Language Models for Link Prediction (Nov 2022)
38. Zha, H., Chen, Z., Yan, X.: Inductive Relation Prediction by BERT. Proceedings of the AAAI Conference on Artificial Intelligence **36**(5), 5923–5931 (Jun 2022)
39. ZHANG, SHUAI., Tay, Y., Yao, L., Liu, Q.: Quaternion Knowledge Graph Embeddings. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
40. Zhao, A., Yu, Y.: Knowledge-enabled bert for aspect-based sentiment analysis. Knowledge-Based Systems **227**, 107220 (2021)