Pretrained Transformers for Simple Question Answering over Knowledge Graphs

Denis Lukovnikov[1], Asja Fischer[2], and Jens Lehmann[1,3]

[1] University of Bonn, Germany
{lukovnik,jens.lehmann}@cs.uni-bonn.de
[2] Ruhr University Bochum, Germany
asja.fischer@rub.de
[3] Fraunhofer IAIS, Dresden, Germany
jens.lehmann@iais.fraunhofer.de

Abstract. Answering simple questions over knowledge graphs is a well-studied problem in question answering. Previous approaches for this task built on recurrent and convolutional neural network based architectures that use pretrained word embeddings. It was recently shown that finetuning pretrained transformer networks (e.g. BERT) can outperform previous approaches on various natural language processing tasks. In this work, we investigate how well BERT performs on SIMPLEQUESTIONS and provide an evaluation of both BERT and BiLSTM-based models in limited-data scenarios.

1 Introduction

Question Answering (QA) over structured data aims to directly provide users with answers to their questions (stated in natural language), computed from data contained in the underlying database or knowledge graph (KG). To this end, a knowledge graph question answering (KGQA) system has to understand the intent of the given question, formulate a query, and retrieve the answer by querying the underlying knowledge base. The task of translating natural language (NL) inputs to their logical forms (queries) is also known as semantic parsing. In this work, we focus on answering simple questions (requiring the retrieval of only a single fact) over KGs such as Freebase [2].

The availability of large quantities of high-quality data is essential for successfully training neural networks on any task. However, in many cases, such datasets can be difficult and costly to construct. Fortunately, the lack of data can be mitigated by relying on transfer learning from other tasks with more data. In transfer learning, (neural network) models are first trained on a different but related task, with the goal of capturing relevant knowledge in the pretrained model. Then, the pretrained model is finetuned on the target task, with the goal of reusing the knowledge captured in the pretraining phase to improve performance on the target task.

Recently proposed transfer learning methods [8,17,18,5,15,10] show that significant improvement on downstream natural language processing (NLP) tasks can be obtained by finetuning a neural network that has been trained for language modeling (LM) over a large corpus of text data without task-specific annotations. Models leveraging these techniques have also shown faster convergence and encouraging results in a few-shot or limited-data settings [8]. Owing to their benefit, the use of this family of techniques is an emerging research topic in the NLP community [10]. However, it has received little attention in KGQA research so far.

The main focus of our work is to investigate transfer learning for question answering over knowledge graphs (KGQA) using models pretrained for language modeling. For our investigation, we choose BERT [5] as our pretrained model used for finetuning, and investigate transfer from BERT using the SIMPLEQUESTIONS [3] task. BERT is a deep transformer [20] network trained on a masked language modeling (MLM) task as well as a subsequent sentence pair classification task. We use SIMPLEQUESTIONS because it is a very well-studied dataset that characterizes core challenges of KGQA, and is, to the best of our knowledge, the largest gold standard dataset for KGQA. The large size of the dataset is particularly appealing for our study, because it allows us to investigate performance for a wider range of sizes of the data used for training. Promising results with BERT for KGQA have been very recently reported on other KGQA datasets [12]. However, we found a thorough investigation of the impact of data availability and an analysis of internal model behavior to be missing, which would help to better understand model behavior in applications of KGQA.

The contributions of this work are as follows:

- We demonstrate for the first time the use of a pretrained transformer network (BERT) for simple KGQA.We also propose a simple change in our models that yields a significant improvement in entity span prediction compared to previous work.
- We provide a thorough evaluation of pretrained transformers on SIMPLEQUESTIONS for different amounts of data used in training and compare with a strong baseline based on bidirectional Long-Short-Term-Memory [7] (BiLSTM). To the best of our knowledge, our work is the first to provide an analysis of performance degradation with reduced training data sizes for SIMPLEQUESTIONS and KGQA in general.
- We try to provide an understanding of the internal behavior of transformer-based models by analyzing the changes in internal attention behavior induced in the transformer during finetuning.

We perform our study using the general framework used in recent works [13,16], where simple question interpretation is decomposed into (1) entity span detection, (2) relation classification, and (3) a heuristic-based post-processing step to produce final predictions. In this work, we particularly focus on the first two subtasks, providing detailed evaluation results and comparison with a baseline as well as [13].

2 Approach

We follow the general approach outlined in BuboQA [13], which decomposes simple question interpretation into two separate learning problems: (1) entity span detection and (2) relation classification. Recent works on SIMPLEQUESTIONS show that this general approach, followed by a heuristic-based entity linking and evidence integration step can achieve state-of-the-art performance [13,16], compared to earlier works [11,4,6,24] that investigated more complicated models.

In summary, our approach follows the following steps at test time:

1. Entity Span Detection and Relation Prediction: The fine-tuned BERT model is used to perform sequence tagging to both (1) identify the span s of the question q that mentions the entity (see Section 2.2) and (2) predict the relation r used in q (see Section 2.3). In the example "*Where was Michael Crichton born?*", s would be the span "*Michael Crichton*". The tagger is trained using annotations automatically generated from the training data and entity labels in Freebase.
2. Entity Candidate Generation: We retrieve entities whose labels are similar to the predicted entity span s using an inverted index[4] and rank them first by string similarity (using `fuzzywuzzy`) and then by the number of outgoing relations. For our example, the resulting set of entity candidates will contain the true entity for Michael Crichton, the writer (corresponding to Freebase URI http://www.freebase.com/m/056wb). Note that the true entity does not necessarily rank highest after the retrieval phase.
3. Query Ranking: Given the relations predicted in Step 1, and the set of entities from Step 2, the entity-relation pairs are re-ranked as detailed in Section 2.4. After ranking entity-relation pairs, we take the top-scoring pair, from which we can trivially generate a query to retrieve the answer from the KG.

Whereas previous works experimented with recurrent and convolutional neural network (RNN resp. CNN) architectures, we investigate an approach based on transformers. Several existing works train separate models for the two learning tasks, i.e. entity span detection and relation prediction. Instead, we train a single network for both tasks simultaneously.

2.1 Background: Transformers and BERT

Transformers: Transformer [20] networks have been recently proposed for NLP tasks and are fundamentally different from the previously common RNN and CNN architectures. Compared to RNNs, which maintain a recurrent state, transformers use multi-head self-attention to introduce conditioning on other timesteps. This enables the parallel computation of all feature vectors in a transformer layer, unlike RNNs, which process the input sequence one time step at a time. And unlike RNNs, which have to store information useful for handling long-range dependencies in its hidden state, the transformer can access any timestep directly using the self-attention mechanism.

More specifically, transformers consists of several layers of multi-head self-attention with feedforward layers and skip connections. Multi-head self-attention is an extension of the standard attention mechanism [1], with two major differences: (1) attention is applied only within the input sequence and (2) multiple attention heads enable one layer to attend to different places in the input sequence.

Let the transformer consist of L layers, each ($l \in \{1,\ldots,L\}$) producing N output vectors $\mathbf{x}_1^{l+1},\ldots,\mathbf{x}_N^{l+1}$, which are then used as inputs in the $l+1$-th transformer layer. The inputs $\mathbf{x}_1^1,\ldots,\mathbf{x}_N^1$ to the first transformer layer are the embeddings of the input tokens $x_1,\ldots,x_T$.

[4]The inverted index maps words to entities whose labels contain that word

The attention scores of the l-th layer are computed as follows:

$$a_{l,h,i,j} = (\mathbf{x}_i^l W_Q^{(l,h)})^\top (\mathbf{x}_j^l W_K^{(l,h)}) \quad , \tag{1}$$

$$\alpha_{l,h,i,j} = \frac{e^{a_{h,l,i,j}}}{\sum_{k=1}^{N} e^{a_{h,l,i,k}}} \quad , \tag{2}$$

where $\alpha_{l,h,i,j}$ is the self-attention score for head $h \in \{1,\ldots,M\}$ in layer l between position i (corresponding to $\mathbf{x}_i^l$) and position j (corresponding to $\mathbf{x}_j^l$) and is implemented as a softmax of dot products between the input vectors $\mathbf{x}_i^l$ and $\mathbf{x}_j^l$, after multiplication with the so called query and key projection matrices for head h of layer l ($W_Q^{(l,h)}$ and $W_K^{(l,h)}$, respectively).

Intermediate representation vectors for each input position are computed as the concatenation of the M heads' summary vectors, each computed as a $\alpha_{l,h,i,j}$-weighted sum of input vectors $\mathbf{x}_1^l,\ldots,\mathbf{x}_N^l$, which are first projected using the matrix $W_V^{(l,h)}$:

$$\mathbf{h}_i^l = [\sum_{j=1}^{N} \alpha_{l,h,i,j} \cdot \mathbf{x}_j^l W_V^{(l,h)}]_{h=1..M} \quad . \tag{3}$$

The output of the l-th transformer layer (which is also the input to the $l+1$-th layer) is then given by applying a two-layer feedforward network with a ReLU activation function on $\mathbf{h}_i^l$, that is:

$$\mathbf{x}_i^{l+1} = \max(0, \mathbf{h}_i^l W_1^{(l)} + b_1^{(l)}) W_2^{(l)} + b_2^{(l)} \quad . \tag{4}$$

For more details, that were omitted here, we refer the reader to the work of Vaswani et al. [20] and other excellent resources, like the Illustrated Transformer[5].

BERT: Following previous work on transfer learning form pretrained transformer-based language models [17], Devlin et al. [5] pretrain transformers on a large collection of unsupervised language data, leading to a model called BERT. However, in contrast to a classical, left-to-right language model used by OpenAI-GPT [17], BERT builds on pretraining a masked language model (MLM). The MLM pretraining is done by randomly masking words, i.e. by randomly replacing them with `[MASK]` tokens, feeding the resulting partially masked sequence into the model and training the model to predict the words that have been masked out, given the other words. This enables BERT's feature vectors to include information both from the preceding tokens as well as the following tokens, whereas the left-to-right LM pretraining of OpenAI-GPT constrained the model to look only at the past. In addition to the MLM task, BERT is also pre-trained on a sentence pair classification task. Specifically, it is trained to predict whether one sentence follows another in a text. This pre-training task is useful for downstream tasks such as entailment, which is formulated as classification of sentence pairs, but also for single sentence classification.

[5] `http://jalammar.github.io/illustrated-transformer/`

BERT for text works as follows. Given a sentence (e.g. "What songs have Nobuo Uematsu produced"), it is first tokenized to (sub)word level using a WordPiece [21] vocabulary (→ ["What", "songs", "have", "no", "#buo", "u", "#ema", "#tsu", "produced"]). More common words are taken as words ("What", "songs", "have"), while uncommon words are split into subword units ("nobuo" → ["no", "#buo"]). This method significantly reduces vocabulary size and the amount of rare words without dramatically increasing sequence length. The input sequence is also padded with a `[CLS]` token at the beginning and a `[SEP]` token at the end.

The WordPiece token sequence is then embedded into a sequence of vectors. Position[6] (and sequence type[7]) embedding vectors are added to the token embeddings. The resulting embedding vectors are fed through the transformer, which uses several layers of multi-head self-attention and feedfoward layers, as described above. The output vectors for each token can be used for sequence tagging tasks, while the vector associated with the `[CLS]` token at the beginning of the sequence is assumed to capture relevant information about the input sequence as a whole, since it has been pre-trained for sentence pair classification.

2.2 Entity span prediction

In this step, we intend to identify the span of tokens in the input question referring to the subject entity mentioned in it. Previous works treated this problem as a binary I/O sequence tagging problem, and explored the use of BiLSTM, conditional random fields (CRFs), and combined BiLSTM-CRF taggers. The sequence tagging model is trained to classify each token in the input sequence as belonging to the entity span (I) or not (O). Instead, we treat span prediction as a classification problem, where we predict the start and end positions of the entity span using two classifier heads. This approach assumes that only one entity is mentioned in the question and that its mention is a single contiguous span. Formally, the start-position classifier has the following form:

$$p(i = \mathtt{START}|x_1, \ldots, x_N) = \frac{e^{\mathbf{x}_i^{L+1\top} \mathbf{w}_{\mathtt{START}}}}{\sum_{j=1}^{N} e^{\mathbf{x}_j^{L+1\top} \mathbf{w}_{\mathtt{START}}}} \quad , \tag{5}$$

where $\mathbf{x}_i^{L+1}$ is the feature vector produced by BERT's topmost (L-th) layer for the i-th token of the sequence and $\mathbf{w}_{\mathtt{START}}$ is the parameter vector of the start position classifier. End position prediction works analogously, applying a different parameter vector, $\mathbf{w}_{\mathtt{END}}$.

2.3 Relation prediction

Relation prediction can be considered a sequence classification task since the SIMPLEQUESTIONS task assumes there is only a single relation mentioned in the question. Thus, for relation prediction, we use BERT in the sequence classification setting where

[6] The use of self-attention requires explicit position indication since this information can not be implicitly inferred, like in RNNs.

[7] BERT uses two sequence types: first-sentence and second-sentence, where the latter is only used for sentence-pair inputs and is thus irrelevant for our task.

we take the feature vector $\mathbf{x}_{\mathsf{CLS}}^{L+1} = \mathbf{x}_1^{L+1}$ produced for the `[CLS]`[8] token at the beginning of the input sequence and feed it through a softmax output layer to get a distribution over possible relations:

$$p(r = R_i | x_1, \ldots, x_N) = \frac{e^{\mathbf{x}_{\mathsf{CLS}}^{L+1\top} \mathbf{w}_{R_i}}}{\sum_{k=1}^{N_R} e^{\mathbf{x}_{\mathsf{CLS}}^{L+1\top} \mathbf{w}_{R_k}}} \quad , \tag{6}$$

where $\mathbf{w}_{R_i}$ is the vector representation of relation R_i[9].

Previous works [16,23,24] propose using the question *pattern* instead of the full original question in order to reduce noise and overfitting. They do this by replacing the predicted entity span with a placeholder token. Doing this would require training a separate model for relation prediction and introduce dependency on entity span prediction. In our BERT-based approach, we chose to train a single model to perform both entity span prediction and relation prediction in a single pass. Thus, we do *not* replace the entity span for relation prediction. We also experimented with training a separate transformer with (1) setting the attention mask for all self-attention heads such that the entity tokens are ignored and (2) replacing the entity mention with a `[MASK]` token. However, both methods failed to improve relation classification accuracy in our experiments.

Training a separate relation classifier network without entity masking yields results equivalent to simply training a single network for both entity span prediction and relation prediction.

2.4 Logical Form Selection

To get the final logical forms, we take the top K (where K=50 in our experiments) entity candidates during entity retrieval and for each, we take the highest-scored relation that is connected to the entity in the knowledge graph. We rank the entity-predicate candidate pairs first based on the string similarity of any of their entity labels/aliases with the identified span, breaking ties by favouring entity-predicate pairs with predicates with higher prediction probability under the BERT model, and the remaining ties are broken by entity in-degree (the number of triples the entity participates in as an object).

3 Experimental Setup

We use the small uncased pretrained BERT model from a PyTorch implementation of BERT[10]. The whole transformer network and original embeddings were finetuned during training. For training, the Adam optimizer [9] was employed and we experimented with different learning rate schedules. Most of the final results reported use a cosine

[8]Before using BERT, the input sequence is first tokenized into WordPieces, a `[CLS]` token at the beginning and a `[SEP]` token is added at the end.

[9]The vector $\mathbf{w}_{R_i}$ is a trainable parameter vector, unique for relation R_i (and is thus not presented by subsymbolic encodings as it is for example the case in [24,11]).

[10]`https://github.com/huggingface/pytorch-pretrained-BERT`

annealing learning rate schedule with a short warmup phase of approximately 5% of total training updates. We indicate if reported results rely on a different schedule.

We used PyTorch 1.0.1 and trained on single Titan X GPU's. The source code is provided at `https://github.com/SmartDataAnalytics/semparse-sq`.

3.1 Metrics

To evaluate the entity span prediction model, we compute the average[11] F1 and span accuracy[12] on word level. Since BERT operates on subword-level (WordPiece), we first need to obtain word-level metrics. To do this, we first transform the predictive distributions over subword units to distributions over words by summing the probabilities assigned to subword units of a word. Then, we take the argmax over the resulting distribution over words.

We also compute F1 on word level over the entire dev/test datasets to compare our numbers to BuboQA [13]. Even though the difference between the dataset-wide F1 and the averaged F1 is small, we believe the latter is more informative, since the contribution of every example is equal and independent of the span lengths.[13] (lower entropy of the predictive categorical distribution)

For relation classification, we report classification accuracy.

3.2 Baseline

As a baseline, we use a BiLSTM start/end classifier for entity span prediction and a BiLSTM sequence classifier for relation prediction. The BiLSTM start/end classifier works on word level and uses the same Glove [14] embeddings as BuboQA [13]. We use the same output layer form as our BERT-based model, where instead of performing a binary I/O tagging of the input sequence, we simply predict the beginning and end positions of the span using a softmax over sequence length (see also Eq. 5). Using this small change significantly improves the performance of our baseline for entity span prediction, as shown in Section 4.

For relation classification, we use a different BiLSTM, taking the final state as the question representation vector and using it in a classifier output as in BuboQA [13] – comprising of an additional forward layer, a ReLU, a batch normalization layer and a softmax output layer. We did not replace the entity mentions with an entity placeholder (like [16]), and instead fed the original sequences into the relation classification encoder.

Even though these BiLSTM baselines are quite basic, previous work has shown they can be trained to obtain state-of-the-art results [13,16].

Both BiLSTMs were trained using a cosine annealing learning rate schedule as the one used to train our BERT-based model.

---

[11] F1, precision and recall are computed separately for each example based on span overlaps and then averaged across all examples in the dev/test set.

[12] Span accuracy is one only for examples where all token memberships are predicted correctly.

[13] The implementation of F1 in BuboQA's evaluation code seems to be computing F1 based on precision and recall computed over the dataset as a whole, thus letting examples with longer spans contribute more towards the final score.

As shown in Section 4, our baselines perform better than or on par with equivalent networks used in BuboQA [13].

3.3 Effect of Limited Training Data

In order to further illustrate the usefulness of fully pretrained models for SIMPLEQUESTIONS and KGQA, we perform a series of experiments to measure how performance degrades when fewer examples are available for training. SIMPLEQUESTIONS is a fairly large dataset containing 75k+ training examples. With abundant training data available, a randomly initialized model is likely to learn to generalize well, which might make the advantage of starting from a fully pretrained model less pronounced. The large size of SIMPLEQUESTIONS makes it possible to study a wider range of limited-data cases than other, smaller datasets.

We run experiments for both BERT and our baseline BiLSTM with different fractions of the original 75k+ training examples retained for training. Examples are retained such that the number of relations not observed during training is minimized, favouring the removal of examples with most frequently occurring relations. We assume that this strategy, compared to random example selection, should not have a big effect on entity span prediction accuracy but should minimize errors in relation prediction due to unseen relation labels, and create more balanced datasets for more informative performance assessment. We report span accuracy and relation accuracy on the validation set of SIMPLEQUESTIONS as a function of the fraction of data retained in Table 3. For relation prediction, we only report experiments where the retained examples cover all relations observed in the full training dataset at least once.

4 Results and Analysis

For the two learning tasks, we observe significant improvements from using BERT, as shown in Table 1a for entity span prediction and Table 1b for relation prediction (see Section 4.1). Section 4.2 talks about experiments with fewer training data, Section 4.3 shows component performance on the test set. Final results for the whole simple QA task are discussed in Section 4.4. Finally, we conclude with an analysis of the attentions in the transformer in Section 4.5.

4.1 Full data results

From Table 1a, we can see that BERT outperforms our BiLSTM baseline by almost 2% accuracy (evaluated on validation set), although the difference in F1 is smaller. Compared to BuboQA [13], we obtain much higher dataset-wide F1 scores, which we attribute to our start/end prediction rather than I/O tagging used by previous works, including BuboQA.

The improvement is less pronounced in relation classification accuracies (see Table 1b), where our baseline BiLSTM achieves the same results as those reported by BuboQA [13] for a CNN. Our BERT-based classifier beats our BiLSTM by almost 1% accuracy.

	Accuracy	Avg. F1	F1*
BiLSTM [13]	–	–	93.1
CRF [13]	–	–	90.2
BiLSTM (ours)	93.8	97.0	97.1
BERT (ours)	95.6	97.8	97.9

(a) Entity span prediction.

	Accuracy
BiGRU [13]	82.3
CNN [13]	82.8
BiLSTM (ours)	82.8
BERT (ours)	83.6

(b) Relation prediction.

Table 1: Component performance evaluation results, trained on all available training data, measured on validation set. (a) Entity span prediction performance, measured by span accuracy, average span F1 and dataset-wide F1 (F1*), all on word level. (b) Relation prediction performance, measured by accuracy (R@1).

Table 2 shows entity retrieval performance for different numbers of candidates, compared against the numbers reported in [13]. The recall at 50 is 2.71% higher. Please note that we also use entity popularity during retrieval to break ties that occur when multiple retrieved entities have the same name (and thus the same string similarity—the main sorting criterion).

4.2 Effect of Limited Training Data

From the limited-data experiments for entity span prediction shown in Table 3 (top part), we can conclude that a pretrained transformer is able to generalize much better with fewer examples. In fact, with only 1% of the original training data used (757 examples), BERT reaches a best span prediction accuracy of 85.4% on the validation set, corresponding to an average F1 of 93.2. In contrast, our BiLSTM baseline achieves only 74.0% span prediction accuracy on the validation set, corresponding to 88.6 F1. In an extremely data-starved scenario, with only 0.03% of the original dataset—corresponding to just 22 training examples—the best validation accuracy we observed for BERT was 62.5%, corresponding to 80.9 F1. In the same setting, we were not able to obtain more than 33.1% accuracy with our BiLSTM baseline. Overall, we can clearly see that the degradation in performance with less data is much stronger for our Glove-based BiLSTM baseline.

R@N	BiLSTM [13]	BiLSTM (ours)	BERT (ours)
1	67.8	76.45	77.17
5	82.6	87.46	88.18
20	88.7	91.47	92.13
50	91.0	93.07	93.71
150	–	94.88	95.40

Table 2: Entity recall on validation set.

		0.03% (22)	0.2% (151)	1% (757)	2.5% (1k9)	5% (3k8)	10% (7k6)	25% (18k9)	50% (37k9)	75% (56k8)	100% (75k7)
Entity Span	BiLSTM	33.1	64.5	74.0	78.1	82.5	85.5	90.1	92.0	93.4	93.8
	BERT	62.5	79.1	85.4	88.9	90.8	92.4	94.2	94.9	95.5	95.6
Relation	BiLSTM	–	–	–	26.5	41.0	56.3	72.4	79.0	81.3	82.8
	BERT	–	–	–	29.6*	48.6	67.5	76.5	80.1	82.6	83.6

Table 3: Entity span detection accuracies (top half) and relation prediction accuracies (bottom half) as a function of fraction of training data retained. Evaluated on the entire validation set. (*) indicates a cosine learning rate schedule with restarts — in extremely low data scenarios for relation classification, this seems to yield better results than the cosine learning rate schedule without restarts that is used everywhere else.

Limited-data experiments for relation prediction (shown in Table 3) (bottom part) reveals that relation classification is more challenging for both our BiLSTM and BERT-based models. However here too, BERT seems to degrade more gracefully than our Glove+BiLSTM baseline.

4.3 Performance on test set

After identifying good hyperparameters for both our BiLSTM baseline and our BERT-based model using the validation set, we evaluated our models using the same evaluation metrics on the test set. Results for both entity span prediction and relation prediction on the test set are reported in Table 4.[14] As shown in Table 4, the test set results are close to the validation set results for both models.

4.4 Final results

In Table 5, we compare our final predictions against previous works on SIMPLEQUESTIONS. With our simple entity linking and logical form selection procedure (see Section 2.4), we achieve 77.3% accuracy on the test set of SIMPLEQUESTIONS, beating

	Entity Span Accuracy	Entity Span Avg. F1	Relation Accuracy
BiLSTM	93.2	96.7	82.4
BERT	95.2	97.5	83.5

Table 4: Component results on test set.

[14]Note that the test set contains “unsolvable” entries, where the correct entity span has not been identified in pre-processing. For these examples, we set the accuracy and F1 to zero.

[15][19] is not included in the comparison because neither [13] or [16] could reproduce the reported results (86.8%).

Approach	Accuracy
MemNN [3]	61.6
Attn. LSTM [6]	70.9
GRU [11]	71.2
BuboQA [13]	74.9
BiGRU [4]	75.7
Attn. CNN [23]	76.4
HR-BiLSTM [24]	77.0
BiLSTM-CRF [16]	78.1
BERT (ours)	77.3

Table 5: Final accuracy for the full prediction task on the test set of SIMPLEQUESTIONS. [15]

all but one of the existing approaches. We suspect that the final score can be further improved by finding better rules for logical form selection, however that is not the goal of this study.

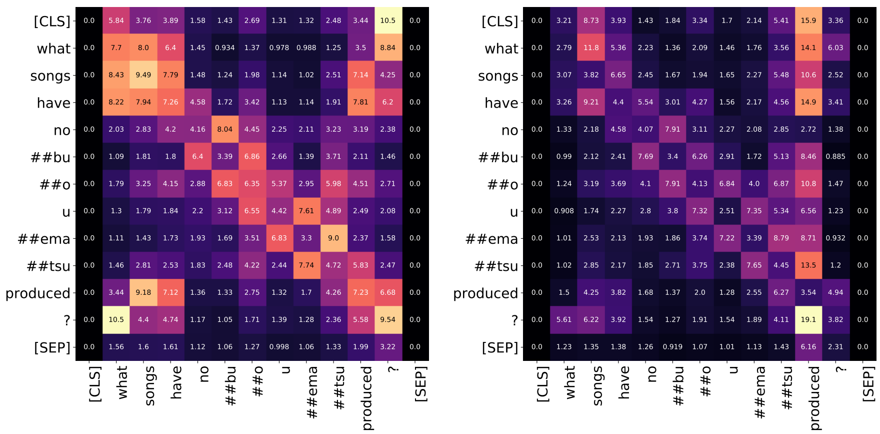
Investigating the entity and relation prediction accuracies separately, we find accuracies of 82.7% for entities and 86.6% for relations. Comparing the 86.6% for relation accuracy after re-ranking (Section 2.4) to the 83.5% (Table 4) relation accuracy before the re-ranking confirms that re-ranking has helped to reduce errors. By analyzing the 22.7% of test examples that were predicted incorrectly, it turned out that in 35% of those cases both a wrong relation and a wrong entity had been predicted, in 41% only the entity was wrong and 24% had only a wrong relation. Of all the cases where the entity was predicted wrong, in 28.6% cases this resulted from the correct entity missing in the candidate set. Entity retrieval errors are also correlated with relation errors: of the cases where the correct entity was not among the retrieved candidates, 71.2% had a wrongly predicted relation, against 55.7% for cases where the correct entity was among the candidates.

4.5 Attention analysis

One of the advantages of using transformers is the ability to inspect the self-attention weights that the model uses to build its representations. Even though this does not completely explain the rules the model learned, it is a step towards explainable decision making in deep learning, and a qualitative improvement upon RNNs. In an attempt to understand how the model works, before and after fine-tuning, we manually inspected the attention distributions used by the transformer network internally during the encoding process.

We compute the average of the 144 attention distributions produced by the $M = 12$ different attention heads in each of the $L = 12$ layers of the employed BERT network:

$$\beta_{i,j} = \frac{\sum_{l=1}^{L} \sum_{h=1}^{M} \alpha_{l,h,i,j}}{L \cdot M} \quad , \tag{7}$$



(a) Before fine-tuning (b) After fine-tuning

Fig. 1: Average attention distribution for the example "*What songs have Nobuo Uematsu produced?*", (a) before training on our tasks (vanilla pretrained BERT) , and (b) after training on our tasks (finetuned BERT). The numbers are scaled to values between 0 and 100, and are computed by averaging of the attention distributions over all heads in all layers, and multiplying the average by 100. We set the scores for `[CLS]` and `[SEP]` tokens to zero in the plots since they always receive a much higher average attention weight than the actual words from the sentence and thus would dominate the plot.

where $\alpha_{l,h,i,j}$ are the attention probabilities as computed in Eq. 2, Here, $\beta_{i,j}$'s are the average attention scores; these values are displayed in Figures 1 and 2 (multiplied by 100 for scaling). More concretly, we compare this average attention signature of a (vanilla) BERT network before fine-tuning it with the attention signature of a BERT model fine-tuned for our tasks (recall that we trained a single model to perform both border detection and relation classification simultaneously). By comparing the attentions before and after training on our tasks, we can identify differences in internal behavior of the model that arose during training.

In Figure 1, the average of all attention distributions is shown for an example question for two versions of the transformer model: pre-trained (vanilla) BERT and BERT fine-tuned for our tasks. While in general, the average attention distribution roughly follows the same patterns after fine-tuning, we can see that the behavior of the attention mechanism responsible for building the representation of the `[CLS]` token is significantly different. We found that, before fine-tuning, the representation building of the `[CLS]` token generally focuses on punctuation and less strongly, on other words. After finetuning, `[CLS]`'s representation is strongly focused on words that characterise the relation conveyed by the sentence. For example, for the question "*Who wrote Gulliver's travels?*" (see Figure 2, first example), the attention is shifted towards the word "*wrote*", which specifies the authorship relationship of the intended answer with the subject en-

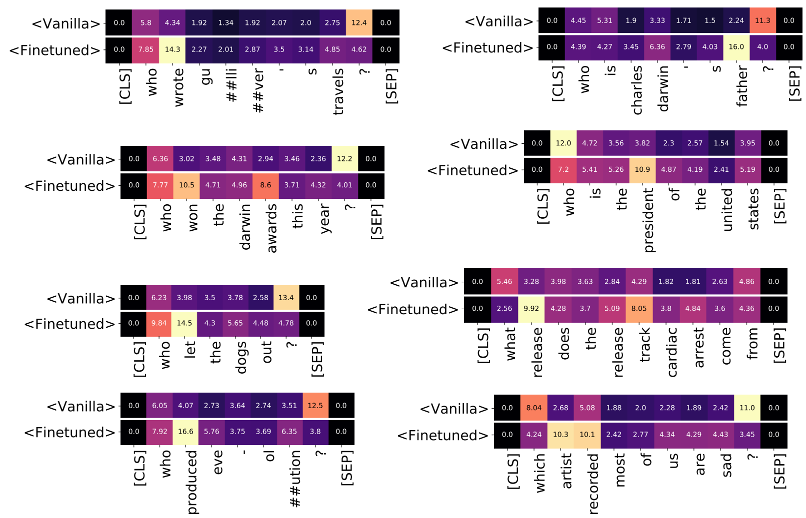


Fig. 2: Average attention distributions for the `[CLS]` token for several examples. `<Vanilla>` is pretrained BERT before finetuning. `<Finetuned>` is BERT finetuned on our tasks. The numbers are scaled to values between 0 and 100, and are computed by averaging of the attention distributions over all heads in all layers, and multiplying the average by 100.

tity (*Gulliver's Travels*) mentioned in the question. We provide several other examples of this kind of attention change in Figure 2.

This change in internal attention behavior can be explained by the fact that sequence classification for relation prediction is done based on the representation built for the `[CLS]` token and attending to relation-specifying words more would produce more useful features for the classifier.

5 Related Work

Bordes et al. [3] propose a memory network (MemNN)-based solution to SIMPLEQUESTIONS. They use bag-of-words representations for triples and question and train a model that predicts the right triple by minimizing a margin-based ranking loss as defined in the section above. They compute scores between questions and whole triples, including the triple objects. However, triple objects are answers and thus might not be present in the question, which may affect the performance adversely. The same work introduces the SIMPLEQUESTIONS dataset, consisting of approximately 100,000 question-answer pairs.

Follow-up works on SIMPLEQUESTIONS typically predict the subject entity and predicate separately, (unlike [3], which ranks whole triples). [6] explore fully character-level encoding of questions, entities and predicates, and use an attention-based decoder [1]. [11] explore building question representations on both word- and character-level. [24] explore relation detection in-depth and propose a hierarchical word-level and symbol-level residual representation. Both [4] and [11] improve upon them by incorporating structural information such as entity type for entity linking. [4] and [23] propose an auxiliary BiRNN+CRF sequence labeling model to determine the span of the entity. The detected entity span is then used for filtering entity candidates. Further, [13] investigates different RNN and CNN-based relation detectors and a BiLSTM and CRF-based entity mention detectors. [16] estimates the upper-bound on accuracy for SIMPLEQUESTIONS, which is less than 83.4% due to unresolvable ambiguities (which is caused by the question lacking information to correctly disambiguate entities). Both [16] and [13] first identify the entity span, similarly to previous works, but disambiguate the entity without using neural networks. With extensive hyperparameter tuning, relatively basic models and simple heuristics, [16] outperformed previous approaches.

[22] proposes a semi-supervised method for semantic parsing based on a structured variational autoencoder that treats logical forms as tree-structured latent variables, and also performs experiments in limited data settings (on ATIS and DJANGO).

6 Discussion

As demonstrated by our experiments, BERT significantly outperforms a strong BiLSTM baseline on the learning problems of relation classification and entity span prediction for simple questions. Moreover, the pre-trained transformer shows less performance decrease when confronted with fewer training data, as can be seen in our limited-data study, which to the best of our knowledge is the first ever conducted for the SIMPLEQUESTIONS data set. The final results on the whole SIMPLEQUESTIONS task are competitive with the current state-of-the-art.

Our comparison of a fully pre-trained transformer to a BiLSTM-based model where only the word embeddings have been pretrained (Glove) might not yield a fair comparison between the two architectures (transformer vs BiLSTM). Further insights could be gained by analyzing the performance of a BiLSTM, which has also been pretrained as a language model (maybe combined with other tasks) in future. Here instead, our aim was to provide evidence that the use of neural networks pre-trained as language model is beneficial for knowledge graph-based question answering like answering SIMPLEQUESTIONS, a usecase not included in the original BERT evaluation and, to the best of our knowledge, not yet explored in the literature.

Even though BERT improves upon our BiLSTM baseline on SIMPLEQUESTIONS, the improvements in the full data scenario might not justify the significantly longer training and inference times and memory requirements. These practical concerns, however, could be mitigated by practical tweaks and future research. Furthermore, with fewer data the performance increases w.r.t. the baseline become more spectacular, indicating that using pre-trained networks like BERT might be essential for achieving reasonable performance in limited data scenarios. Such scenarios are common for datasets with more complex questions. Therefore, we believe pretrained networks like BERT can have a bigger impact for complex KGQA (even when training with all data available).

In the future, we plan to provide a better investigation of BERT's decision process and its limitations. We also plan to investigate the robustness of existing methods as well as BERT-based models to typing mistakes and variation in expressing questions (e.g. using synonyms).

Acknowledgement

We acknowledge support by the European Union H2020 Framework Project Cleopatra (GA no. 812997).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250. AcM (2008)
3. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
4. Dai, Z., Li, L., Xu, W.: Cfo: Conditional focused neural question answering with large-scale knowledge bases. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 800–810 (2016)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. He, X., Golub, D.: Character-level question answering with attention. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1598–1607 (2016)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 328–339 (2018)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Liu, X., He, P., Chen, W., Gao, J.: Multi-task deep neural networks for natural language understanding. arXiv preprint arXiv:1901.11504 (2019)
11. Lukovnikov, D., Fischer, A., Lehmann, J., Auer, S.: Neural network-based question answering over knowledge graphs on word and character level. In: Proceedings of the 26th international conference on World Wide Web. pp. 1211–1220. International World Wide Web Conferences Steering Committee (2017)
12. Maheshwari, G., Trivedi, P., Lukovnikov, D., Chakraborty, N., Fischer, A., Lehmann, J.: Learning to rank query graphs for complex question answering over knowledge graphs. In: International Semantic Web Conference. Springer (2019)
13. Mohammed, S., Shi, P., Lin, J.: Strong baselines for simple question answering over knowledge graphs with and without neural networks. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). vol. 2, pp. 291–296 (2018)
14. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

15. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL (2018)
16. Petrochuk, M., Zettlemoyer, L.: Simplequestions nearly solved: A new upperbound and baseline approach. arXiv preprint arXiv:1804.08798 (2018)
17. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training
18. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Tech. rep.
19. Ture, F., Jojic, O.: No need to pay attention: Simple recurrent neural networks work! In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 2866–2872 (2017)
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
21. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
22. Yin, P., Zhou, C., He, J., Neubig, G.: Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. pp. 754–765. Association for Computational Linguistics (2018), `https://aclanthology.info/papers/P18-1070/p18-1070`
23. Yin, W., Yu, M., Xiang, B., Zhou, B., Schütze, H.: Simple question answering by attentive convolutional neural network. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 1746–1756 (2016)
24. Yu, M., Yin, W., Hasan, K.S., dos Santos, C., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 571–581 (2017)