EEE-QA: Exploring Effective and Efficient Question-Answer Representations

Zhanghao Hu* Yijun Yang* Junjie Xu* Yifu Qiu Pinzhen Chen
School of Informatics, University of Edinburgh
huzh666295@gmail.com thomasyyj@outlook.com smyjx1@163.com
yifu.qiu@ed.ac.uk pinzhen.chen@ed.ac.uk

Abstract

Current approaches to question answering rely on pre-trained language models (PLMs) like RoBERTa. This work challenges the existing question-answer encoding convention and explores finer representations. We begin with testing various pooling methods compared to using the begin-of-sentence token as a question representation for better quality. Next, we explore opportunities to simultaneously embed all answer candidates with the question. This enables cross-reference between answer choices and improves inference throughput via reduced memory usage. Despite their simplicity and effectiveness, these methods have yet to be widely studied in current frameworks. We experiment with different PLMs, and with and without the integration of knowledge graphs. Results prove that the memory efficacy of the proposed techniques with little sacrifice in performance. Practically, our work enhances 38–100% throughput with 26–65% speedups on consumer-grade GPUs by allowing for considerably larger batch sizes. Our work sends a message to the community with promising directions in both representation quality and efficiency for the question-answering task in natural language processing.

Keywords: Question Answering, Representation Learning, Memory Efficiency

1. Introduction

Knowledge base question answering (KBQA) performs question answering (QA) using a knowledge base (KB) as its primary source of information (Mihaylov et al., 2018; Talmor et al., 2019; Lan et al., 2021). Popular KBQA approaches use a text encoder and a graph neural network (GNN) to derive representations for question-answer pairs and a knowledge base, followed by joint reasoning over their representations (Wang et al., 2022; Zhang et al., 2022; Sun et al., 2022; Hao et al., 2022; Yasunaga et al., 2021).

Our motivation arises from two limitations in these works: (1) **inference efficiency**: the majority of QA models frame answer predictions as scoring to which degree a candidate answer can satisfactorily respond to a given question. Consequently, for each question, a model must perform inference as many times as there are candidates. We suggest an alternative approach that encodes all candidates alongside the question and trains the model to discern the most probable answer. This formalism enhances inference efficiency too since the QA model considers all candidates simultaneously in one go. (2) **semantic modelling for QA**: the PLM begin-of-sentence token, i.e. CLS, is commonly used as the representation for an input question-answer pair for QA prediction or KG interaction; however, it has long been argued that this representation is suboptimal in capturing the input semantics (Reimers and Gurevych, 2019).

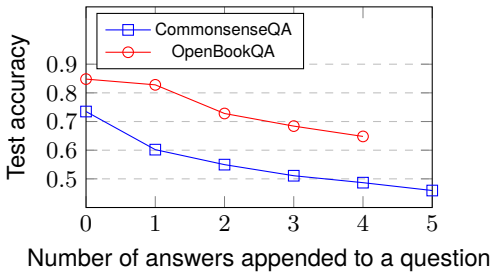

Figure 1: A pilot experiment of appending random answers to the question before performing QA. Tested on CommonsenseQA (5 candidates) and OpenBookQA (4 candidates) with GreaseLM.

In this work, we first demonstrate that it is non-trivial for the QA model to infer a question with multiple candidates in a single pass. To address this challenge, we revisit the representation for input QA pairs and show that by simply pooling we can improve models' performance by a considerable margin. We then propose to delay the question-answer concatenation step to after a single-pass PLM encoding of the question and all answer candidates for improved memory efficiency. Experiments show that max pooling brings in substantial gains, even exceeding the current state-of-the-art KBQA models. On top of this, our proposed structure maintains a similar performance to the baseline while incurring less computation thus improving throughput. Our code is publicly available.[1]

*The first three authors contributed equally. Author ordering has been determined by coin flip.

[1] https://github.com/Thomasyyj/EEEQA

2. From One to Multiple Answers

Intuitively, having all answer candidates together with the question in a single read may help models capture the nuances between them so as to make a better choice. Illustratively, it would be easier to cope with 2) than 1) below:

1) Seeing "*What do you do if you're late for an appointment? Cancel the appointment.*" and separately seeing "*What do you do if you're late for an appointment? Apologize and explain.*"

2) Answering "*What do you do if you're late for an appointment? Cancel the appointment, or Apologize and explain.*"

However, most KBQA works focus on modelling a score for each QA pair without inter-answer considerations. This also leads to larger memory usage when a system is deployed, as the question needs to be repeatedly encoded with each choice. A straightforward way to let the model access the information from all other answer candidates is to add all answer candidates to the question at the encoding stage. We perform a pilot test by incrementally and randomly appending answers from the candidate set to the end of a question and then use GreaseLM (Zhang et al., 2022) to perform QA in the conventional style. We leave the technical description to Section 3.3. As Figure 1 shows, with more answer choices added to the question, the system performance decreases monotonically. This reflects that naively dealing with multiple answers at the same time for KBQA will undermine the judgment of a system. We are therefore motivated to search for more feasible solutions.

3. Methodology

3.1. Preliminaries

With a question in discrete tokens $Q = [x_1, x_2, ..., x_{|Q|}]$ and a set of n answer choices $\mathcal{A} = \{A_1, A_2, ..., A_n\}$, current methods use a PLM $f()$ to encode the question and each A_i as a pair $(Q, A_i)_{A_i \in \mathcal{A}}$ into representations $\mathcal{H} = [H_{\texttt{<s>}}, H_{x_1}, ..., H_{x_{|Q|}}, H_{\texttt{</s>}_1}, H_{\texttt{</s>}_2}, H_{A_i}, H_{\texttt{</s>}_3}] = f([\texttt{<s>}, Q, \texttt{</s>}_1, \texttt{</s>}_2, A_i, \texttt{</s>}_3])$, where <s> and </s> denote begin-of-sentence (or CLS) and end-of-sentence (or SEP) tokens respectively. This is carried out separately for each A_i for a total of n times. A multi-layer perception $g()$ then computes a score using the concatenation of the question CLS token and the answer embedding $S_i = g(H_{\texttt{<s>,i}} \oplus H_{A_i})$ for each (Q, A_i) pair. The final answer A_i is selected by $i = argmax(S_1, S_2, ..., S_n)$. In this way, the PLM encodes the question with each answer candidate in n passes, which we refer to as **1AnP** encoding.

3.2. Improved question representations

In previous QA works, the question representation is used for the interaction with a KG (if included) and for score prediction, but the resulting $H_{\texttt{<s>}}$ from the conventional CLS pooling cannot capture the full semantics of the input (Reimers and Gurevych, 2019). To seek a more meaningful representation, we propose a pooling operation $pool()$ over all question token embeddings. Having the same QA representations $\mathcal{H}$, the score for (Q, A_i) can then be computed as $S_i = g(pool(H_{\texttt{<s>}}, H_{x_1}, ..., H_{x_{|Q|}}, H_{\texttt{</s>}_1}) \oplus H_{A_i})$ instead. Our work investigates four different pooling operations:

- **Max pooling** selects the maximum element in each dimension from all embeddings.
- **Mean pooling** averages all embeddings.
- **Attentive pooling** sums up all embeddings, weighted by a learnable attention vector.
- **Layerwise CLS pooling** adds up all the begin-of-sentence embeddings from each PLM layer, weighted by a learnable attention vector (Tenney et al., 2019).

3.3. Single-pass encoding

The conventional question-answer encoding is memory inefficient because the often long question needs to be encoded multiple times with different answers. We propose a one-pass technique, termed **nA1P**, where a PLM encodes a question and all candidate answers simultaneously to reduce memory usage. This also resembles how a human tackles a multiple-choice question: answers are compared to each other before a decision is made. We append all candidates to the question and embed with $f()$ in one pass: $[\texttt{<s>}, Q, \texttt{</s>}_1, \texttt{</s>}_2, A_1, \texttt{</s>}_3, A_2 ..., A_n, \texttt{</s>}_{n+2}]$.

To alleviate the potential interference between answers when they are being encoded altogether as illustrated in Section 2, we propose two levels of semantic integration by comparing and fusing the answer representations and then merging them with the question. For the answer comparison, inspired by prior research which runs a gate mechanism over answers in multiple encoding passes (Zhang et al., 2020; Hao et al., 2022), we propose a gated interaction adapted to our single-pass scheme.

Specifically, we first perform the aforementioned pooling on the entire span of each answer token as its representation $H_{A_i} = pool(H_{\texttt{</s>}_{i+1}}, H_{A_i}, H_{\texttt{</s>}_{i+2}})$. For H_{A_i} and all other answers $H_{A_j, j \neq i}$, we then compute a multi-head attention (MHA) score α_{ji} of H_{A_j} queried by H_{A_i} after pooling. Intuitively, this allows the explicit comparison between answer A_i and each other answer A_j. We derive the final answer representation $\hat{H}_{A_i}$ by applying a gate mechanism to balance the information from answers H_{A_i} and H_{A_j} using the

attention score α_{ij} computed as:

$$\alpha_{ji} = \text{MHA}(\text{query}=H_{A_i}, \text{key}=H_{A_j}, \text{value}=H_{A_j})$$
$$\hat{H}_{A_i} = \gamma H_{A_i} + (1-\gamma)(\textstyle\sum_j \alpha_{ji} H_{A_j})$$

where $\gamma = \sigma(W_1 H_{A_i} + W_2(\sum_j \alpha_{ji} H_{A_j}) + bias)$ is a weight to balance the information from a particular answer and other answers, $\sigma()$ is a sigmoid function, and W_1, W_2 are trainable matrices. Question-answer scoring remains similar to Section 3.2; we concatenate the pooling outcome of the question and each answer's gated representation to score each (Q, A_i) pair: $S_i = g(pool(H_{\langle s \rangle}, H_{x_1}, ..., H_{x_{|Q|}}, H_{\langle /s \rangle_1}) \oplus \hat{H}_{A_i})$.

In addition, we formally define our setting in Section 2 as **nAnP**, which affixes all multiple candidates to the original question to form an extended question, and then encodes the modified question with each candidate A_i as a pair $([Q, A_1, ..., A_n], A_i)_{A_i \in \mathcal{A}}$ into representations $\mathcal{H}$ for n times separately for each A_i. The scoring criterion to select the final answer is similar to Section 3.2. This nAnP technique bridges the conventional approach and our single-pass scheme, and creates a fair comparison between 1AnP and nA1P with controlled variations.

4. Experiments and Discussions

4.1. Data and evaluation

We evaluate our approaches on two QA datasets. First, **CommonsenseQA** (Talmor et al., 2019) contains 12,102 questions, each paired with five answer candidates; the questions demand commonsense knowledge. We follow the in-house data split by Lin et al. (2019) because the official test is not public. Next, **OpenBookQA** (Mihaylov et al., 2018) comprises 5,957 4-way multiple-choice questions related to elementary scientific knowledge. For this dataset, we follow the official data splits. **Accuracy** (%) is used as the metric.

4.2. Experimental setup

Pooling We first explore the best pooling strategy based on GreaseLM (Zhang et al., 2022) since it uses the CLS token for both KG interaction and QA scoring, but we consider this suboptimal. To position our pooling results in current research, we include several prior works for comparison.

Efficient inference We compare our proposed methods: single-pass inference nA1P, and the vanilla scheme nAnP from Section 3.3 with the conventional 1AnP. We extensively test them in three scenarios: PLM-only (RoBERTa-Large, Liu et al., 2019b), PLM with KG (PLM+KG, Yasunaga et al., 2021), as well as PLM with KG and an interaction node (PLM+KG+Int, Zhang et al., 2022).

System	Accuracy (std.)
RoBERTa-Large (Liu et al., 2019b)	68.69 (±0.56)
QA-GNN (Yasunaga et al., 2021)	73.41 (±0.92)
JointLK (Sun et al., 2022)	74.43 (±0.83)
ACENet (Hao et al., 2022)	74.72 (±0.70)
GreaseLM (Zhang et al., 2022)	74.20 (±0.40)
GreaseLM (Ye et al. (2023)'s re-run)	73.60 (unknown)
GreaseLM (our re-run)	73.57 (±0.08)
+ mean pooling	73.73 (±0.29)
+ max pooling	75.42 (±0.52)†
+ attentive pooling	73.97 (±0.51)
+ layerwise CLS pooling	73.97 (±0.16)

Table 1: Performance (%, accuracy from 3 runs) of pooling techniques compared with previous works on CommonsenseQA. †Significantly better than GreaseLM and other pooling methods with p-values < 0.05 in pairwise t-tests.

Hyperparameters Our PLM backbone is RoBERTa (Liu et al., 2019b). In experiments involving a KG, we use ConceptNet (Xu et al., 2021). GNN node embeddings are initialized with entity embeddings from Feng et al. (2020). For each QA pair, we retrieve the top 200 nodes and their adjacent edges based on node relevance scores following Xu et al. (2021). We use a dimension of 200 and 5 GNN layers, with a 0.2 dropout probability. We use batch sizes 64 and 128 for CommonsenseQA and OpenBookQA correspondingly. We train all models with the RAdam optimizer (Liu et al., 2019a) on a single Nvidia RTX A5000. Learning rates for the PLM parameters and non-PLM modules are 1e-5 and 1e-3 separately. We determined these hyperparameters using the development set. We note that we use a slightly different but necessary implementation in the PLM+KG+Int experiments with our nA1P framework since the information interaction requires an additional special token for each answer.

4.3. Pooling results

We first report the performance of our proposed pooling methods in Table 1. We see improvements in most pooling methods on the GreaseLM architecture, showing the usefulness of a finer representation. Among all, max pooling demonstrates an incredible gain, even outperforming current SOTA models. This highlights that a simple but effective representation has long been neglected in the community. We therefore stick to max pooling as opposed to CLS pooling for subsequent experiments.

System	Pooling	CommonsenseQA			OpenBookQA		
		PLM	+ KG	+ KG + Int	PLM	+ KG	+ KG + Int
1AnP (previous)	CLS	70.02	72.82	73.97	80.20	81.80	81.60
	Max	70.51	73.41	75.42	82.40	82.40	82.60
nAnP (contrastive)	CLS	67.12	69.62	68.82	79.40	78.40	81.80
	Max	67.12	68.90	69.14	79.40	82.60	82.20
nA1P (ours)	CLS	67.77	69.30	69.38	78.80	79.40	80.20
	Max	68.25	68.65	70.91	79.00	80.60	80.40
	Max + Gate	69.62	71.88	70.91	79.60	80.60	81.40

Table 2: Performance (accuracy, %) of our systems on CommonsenseQA and OpenBookQA.

GPU Model	Mem. (GB)	Batch size (↑)		Inference time (↓)	
		1AnP	nA1P (Δ%)	1AnP	nA1P (Δ%)
RTX A5000	24	100	160 (+60%)	4.61s	3.31s (-28%)
RTX 3090	24	100	160 (+60%)	2.45s	1.82s (-26%)
RTX 2080 Ti	11	30	45 (+50%)	1.13s	0.76s (-33%)
GTX 1080 Ti	11	30	45 (+50%)	2.64s	1.27s (-52%)
Titan X Pascal	12	40	55 (+38%)	3.56s	1.55s (-56%)
GTX 1080	8	10	20 (+100%)	2.21s	0.77s (-65%)

Table 3: Comparison in efficiency between 1AnP and our nA1P: usable batch size and total inference time when solving 1000 QA instances on a single GPU.

4.4. From multiple to one pass

The results from our memory-efficient experiments across two datasets are presented in Table 2.

Pooling For both CommonSenseQA and OpenBookQA, we observe a better performance with max pooling compared to CLS pooling in most of the settings, indicating the effectiveness and generalizability of pooling in QA representation. We note that max pooling is slightly behind when multiple answers are encoded simultaneously and when a KG is used. We conjecture that this is because, with many answers, it is difficult for the information in LMs to align with the corresponding KG sub-graphs.

Efficient inference For our proposed nA1P encoding, results indicate that when seeking improved memory efficiency, the baseline CLS pooling accuracies on both CommonsenseQA and OpenBookQA are sacrificed to a slight degree. However, this can be mitigated through our proposed techniques: max pooling as well as the gated answer representation mechanism. We observe on par if not higher performance when these are added, highlighting the effectiveness of our explicit inter-answer interaction.

Transferability Additionally, we switch the PLM from RoBERTa to BERT (Devlin et al., 2019) in order to investigate whether our nA1P framework is compatible with other PLM resources. We find that the test accuracies are 50.68% for our nA1P approach and 49.80% for the traditional 1AnP. The pattern is the same as we have observed on CommonsenseQA. Nonetheless, the results with BERT as the PLM are significantly lower than RoBERTa.

4.5. Throughput and inference time

To study the practicality of memory efficiency, we run GreaseLM with both 1AnP and our nA1P using the same configurations to solve 1000 QA instances. Table 3 reports the maximum usable batch size and total inference time on a single consumer-grade GPU. To eliminate variable factors like input lengths, we duplicate a single data instance to fill up a batch.

Compared with the baseline 1AnP, while the incorporation of the gated mechanism to our nA1P approach could introduce extra processing time and memory, we highlight both an increase in maximum batch size and a decline in inference time with our method. This framework enables the utilization of larger batch sizes across multiple GPU models which markedly increases throughput. This is particularly advantageous when conducting QA tasks at a large scale.

4.6. Ablation study

For our proposed nA1P approach, we systematically analyze each its component's impact on the performance on the CommonsenseQA in-house test. We eliminate a single module at a time as

System	Accuracy
Max Pool + Q⊕A + Gate	71.88
Max Pool + Q⊕A	69.54
Max Pool	68.65

Table 4: Performance of our nA1P with ablations on CommonsenseQA.

shown in Table 4. Introducing the QA concatenation yields a 0.89% improvement, demonstrating the value of considering both questions and answer contexts for enhanced interaction and reasoning. Notably, having the gate layer module leads to a significant performance boost of 2.34%, emphasizing the crucial role of diverse choice interactions during the message-passing process.

5. Related Work

Knowledge-base Question Answering necessitates models to jointly reason over both external knowledge graphs and parametric knowledge from language models (Talmor et al., 2019; Mihaylov et al., 2018). However, grounding knowledge from both textual and structured modalities presents a non-trivial challenge (Yasunaga et al., 2021; Sun et al., 2022; Zhang et al., 2022). Furthermore, existing works predominantly adhere to the 1AnP setting, which needs multiple inferences to answer a question (Liu et al., 2019b; Hao et al., 2022; Yasunaga et al., 2022).

The only exception is a recent work using decoder-only large language models, where a question and all possible answers can be framed as a natural language input (Robinson and Wingate, 2023). We highlight that our multi-pass encoder models, with similar performance, are significantly more affordable in practice. Finally, Section 2 shows that vanilla nA1P notably lags behind 1AnP, which motivates us to revisit the representation scheme in current models.

6. Conclusion and Outlook

This paper presents our efforts in searching for an effective and memory-efficient representation for question-answering. Experimental results show that a simple max pooling mechanism consistently outperforms CLS pooling which has been widely used in this field. We then propose and test a gated single-pass inference approach to encourage answer interactions and improve efficiency. It is seen that the approach reduces memory usage with a tiny sacrifice in performance. Our paper calls for future work to shift to max pooling and encourages further exploration of efficiency-aware QA representations.

Our investigation focused on the PLM side of question answering, but the KG side remains unoptimized. In our experiments, sub-KGs are constructed for each QA pair, without adjusting for a single-pass encoding scheme where all answers are encoded simultaneously. Thus, although we could encode a question with all answers in our proposed nA1P inference approach, processing separate sub-KGs would slow down the inference. The challenge exists in combining all answer candidates' information into one knowledge graph which can be looked at by future research.

Acknowledgements

This work originated from a course project at the University of Edinburgh. We must thank the course organizers and teaching staff. We are grateful to the anonymous reviewer who identified an oversight in Section 3.1.

Pinzhen Chen has received funding from UK Research and Innovation under the UK government's Horizon Europe funding guarantee [grant numbers 10039436 and 10052546].

Bibliographical References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *EMNLP*.

Chuzhan Hao, Minghui Xie, and Peng Zhang. 2022. ACENet: Attention guided commonsense reasoning on hybrid knowledge graph. In *EMNLP*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *IJCAI*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *EMNLP-IJCNLP*.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. In *ICLR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*.

Joshua Robinson and David Wingate. 2023. Leveraging large language models for multiple choice question answering. In *ICLR*.

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2022. JointLK: Joint reasoning with language models and knowledge graphs for commonsense question answering. In *NAACL*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *ACL*.

Kuan Wang, Yuyu Zhang, Diyi Yang, Le Song, and Tao Qin. 2022. GNN is a counter? revisiting GNN for question answering. In *ICLR*.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2021. Fusing context into knowledge graph for commonsense question answering. In *ACL-IJCNLP Findings*.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *NeurIPS*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *NAACL*.

Qichen Ye, Bowen Cao, Nuo Chen, Weiyuan Xu, and Yuexian Zou. 2023. FiTs: fine-grained two-stage training for knowledge-aware question answering. In *AAAI*.

Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2020. DCMN+: Dual co-matching network for multi-choice reading comprehension. In *AAAI*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *ICLR*.