QAVSA: Question Answering using Vector Symbolic Algebras

Ryan Laube and **Chris Eliasmith**
University of Waterloo, ON, Canada
{rlaube,celiasmith}@uwaterloo.ca

Abstract

With the advancement of large pretrained language models (PLMs), many question answering (QA) benchmarks have been developed in order to evaluate the reasoning capabilities of these models. Augmenting PLMs with external knowledge in the form of Knowledge Graphs (KGs) has been a popular method to improve their reasoning capabilities, and a common method to reason over KGs is to use Graph Neural Networks (GNNs). As an alternative to GNNs to augment PLMs, we propose a novel graph reasoning module using Vector Symbolic Algebra (VSA) graph representations and a k-layer MLP. We demonstrate that our VSA-based model performs as well as QA-GNN, a model combining a PLM and a GNN-module, on 3 multiple-choice question answering (MCQA) datasets. Our model has a simpler architecture than QA-GNN and also converges 39% faster during training.

1 Introduction

Models that perform question answering tasks require some amount of knowledge, whether it is structured or implicit, about the concepts to be reasoned over. Modern large pretrained language models (PLMs), linguistic knowledge is implicit in the token embeddings that have been learned using a self-attention mechanism on large text corpuses to perform next-token prediction (Vaswani et al., 2017). With enough model parameters and training data, these PLMs have been successful in a wide range of question-answering and reasoning benchmarks. However, when analyzing deductive reasoning performance, i.e. the ability to learn and generalize from logic rules, smaller PLMs like BERT and RoBERTa demonstrate inconsistent performance (Yuan et al., 2023).

As a result, there have been many studies that work to integrate external, structured knowledge and reasoning modules with PLMs to perform more reliable logical reasoning. One type of knowledge structure that is commonly used are knowledge graphs (KG), due to their suitability for symbolic reasoning (Lan et al., 2021). Graph Neural Networks (GNNs) are deep learning networks that have gained popularity for reasoning over graph-structured data. Consequently, methods that integrate KGs with PLMs often also use GNNs (Ye et al., 2022). One recent approach to combining these techniques is captured by the QA-GNN (Yasunaga et al., 2021), which is able to answer multiple-choice questions by scoring the plausibility of each question answer.

Rather than using GNNs to model structured data, our model, QAVSA, uses a lesser known method, Vector Symoblic Algebras (VSAs; also known as Vector Symbolic Architectures), to represent and combine high-dimensional concept vectors in a structured way. The integration of PLMs and VSAs has not been previously proposed, to our knowledge. As a result, the focus of this paper is to perform a comparison between our VSA-based method and the GNN-based method of QA-GNN for improving reasoning capabilities of PLMs in the context of multiple-choice question answering.

Specifcally, we compare the performance of QAVSA and QA-GNN on CommonsenseQA (Talmor et al., 2019), OpenbookQA (Mihaylov et al., 2018), and MedQA-USMLE (Jin et al., 2021), with the first two datasets being focused on commonsense reasoning, and the latter focusing on domain-specific medical questions.

The main contributions of this paper are: 1) a novel combination of PLM text encoders and VSAs that performs as well as an analgous GNN-based method on three MCQA datasets while training faster; 2) a comparative evaluation of different VSAs and PLMs used in the model; 3) a study on ablation and variations of the model architecture and graph embedding representations; and 4) a new VSA-specific method of analyzing model

Proceedings of the 9th Workshop on Representation Learning for NLP (RepL4NLP-2024), pages 191–202

explainability.

2 Related Work

Many of the top performing models on MCQA benchmarks involving reasoning are larger 10B or 100B+ parameter PLMs. High performance can come from fine-tuning on MCQA datasets as has been shown with UnifiedQA (Khashabi et al., 2020) and UNICORN Lourie et al. (2021). High performance with these models has also been shown to be achievable with prompting techniques including few-shot prompting (Anil et al., 2023), Chain-of-Thought prompting with self-consistency (Huang et al., 2023a), and ensemble refinement (Singhal et al., 2023).

More relevant to our research, several approaches have been proposed to integrate external knowledge graphs with PLMs in order to make use of more domain-specific structured knowledge. These include KEAR (Xu et al., 2022) and DEKCOR (Xu et al., 2021), which use a PLM and knowledge retrieved from an external KG and Wiktionary to train an attention mechanism on these external knowledge bases and PLM representations to improve commonsense reasoning.

Similarly, KagNet (Lin et al., 2019) performs commonsense reasoning by grounding the concepts within each question-answer (QA) pair of multiple-choice datasets to extract subgraphs from an external knowledge graph. KagNet then uses a combination of Graph Convolutional Networks, LSTM-based relational path encodings, and a path-based attention mechanism to identify important reasoning paths to generate graph vector encodings for each QA pair to subsequently score them. Adopting similar graph preprocessing to KagNet, MHGRN (Feng et al., 2020) performs multi-hop, multi-relation reasoning by using multi-hop message passing from Relational Graph Convolutional Networks, structured relational attention, and node attention pooling to generate its graph representations and score QA pairs.

Another family of models that also use the same graph processing above stem from the QA-GNN (Yasunaga et al., 2021) model. QA-GNN fuses a PLM representation of the QA context as a node into the QA subgraphs. Using a Graph Attention Network (GAT), QA-GNN updates the subgraph concept embeddings, including the QA context node and edge weights. The initial PLM QA context representation, along with the final graph representation of the QA context and graph concept attention pooling is used to score the QA pairs. This method is refined in GreaseLM (Zhang et al., 2021) in which the PLM token and graph node modalities of the QA context are mixed over several layers to simultaneously update the PLM and GNN concept embeddings. Further refinements are made in DRAGON (Yasunaga et al., 2022a), where the cross-modal encoder from GreaseLM is pre-trained in a self-supervised fashion to perform both masked language modeling and KG link prediction. Our QAVSA model uses similar pre-processing to QA-GNN but with a different representation of the graph. Consequently, most of our direct comparisons are to QA-GNN.

There exist other models that use external KGs but are not GNN-based, such as GSC (Wang et al., 2022) and MVP-Tuning (Huang et al., 2023b). GSC uses a simple graph neural counter to reduce the node and embeddings to one dimension and performs GNN-like embedding updates on these single values. MVP-Tuning makes use of semantically similar QA pairs in the training set to improve knowledge retrieval and tunes prompt tokens of the PLM by using the QA context and retrieved KG triplets as input to the PLM.

3 Vector Symbolic Algebras

Vector Symbolic Algebras (VSAs) are defined by a set of three vector operations that are useful for building up structured vector representations. These include the similarity, bundling (or collecting), and binding operations. There are a wide variety of possible choices for these operations, but we consider only two sets of operators, those for Holographic Reduced Representations (Plate, 1995) and for Vector-derived Transformation Bindings (Gosmann and Eliasmith, 2019).

A similarity operation is necessary to compare different vectors within the VSA space, and it is often computed with a normalized vector dot product, i.e., cosine similarity. For two VSA-encoded vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, this is defined as $s(\mathbf{x}, \mathbf{y}) = \frac{<\mathbf{x},\mathbf{y}>}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$. Both HRRs and VTBs use this similarity operator.

A bundling operation is used to represent a set of objects and is usually defined by element-wise addition. Thus, the bundling of $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ can by defined by $S(\mathbf{x}, \mathbf{y}) = \mathbf{x} + \mathbf{y}$. In a VSA, this operation should result in a new vector that is similar to both $\mathbf{x}$ and $\mathbf{y}$, as is the case with element-wise addition. Both HRRs and VTBs use this bundling

operator.

A binding operation, is used to combine two symbols together in a single representation, which is often used to represent slot-filler pairs. In HRRs, circular convolution is used as a binding operator, defined by

$$(\mathbf{x} \circledast \mathbf{y})_i := \sum_{j=1}^{d} x_j y_{((i-j) mod\, d)+1}, i \in \{1, 2, ..., d\}.$$

A desired property of a binding operator is that the resulting vector from $\mathbf{x} \circledast \mathbf{y}$ should be dissimilar to both $\mathbf{x}$ and $\mathbf{y}$. Also, an unbinding operation, or a pseudo-inverse to binding should exist. With circular convolution, this is done by binding the pseudo-inverse of the given operand: $(\mathbf{x} \circledast \mathbf{y}) \circledast^{-1} \mathbf{y} \approx \mathbf{x} \circledast \mathbf{y} \circledast \mathbf{y}^+ \approx \mathbf{x}$. For HRRs the approximate inverse to $\mathbf{y}$ is $\mathbf{y}^+ := (y_1, y_d, y_{d-1}, ..., y_2)^\top$ (Plate, 1995).

Since circular convolution is commutative, there is no directional relation to two bound vectors in the HRR VSA. In contrast, VTB has a non-associative and non-commutative binding operation defined on vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Specifically, given $d^{\frac{1}{2}} = d' \in \mathbb{N}_{>0}$, we have

$$\mathbf{x} \circledast \mathbf{y} := \mathbf{V}_\mathbf{y}\mathbf{x} = \begin{bmatrix} \mathbf{V}'_\mathbf{y} & 0 & 0 \\ 0 & \mathbf{V}'_\mathbf{y} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \mathbf{x}$$

, where

$$\mathbf{V}'_\mathbf{y} = d^{\frac{1}{4}} \begin{bmatrix} y_1 & y_2 & \cdots & y_{d'} \\ y_{d'+1} & y_{d'+2} & \cdots & y_{2d'} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d-d'+1} & y_{d-d'+2} & \cdots & y_d \end{bmatrix}.$$

The approximate inverse to $\mathbf{y}$ is $\mathbf{y}^+$, where the elements of $\mathbf{y}$ are permuted such that $\mathbf{V}_{\mathbf{y}^+} = \mathbf{V}_\mathbf{y}^\top$.

The VTB binding operation has only right inverses and identities, so there exists an alternative Transposed VTB (TVTB) algebra, with two-sided inverses and identities with the following binding operation:

$$\mathbf{x} \circledast \mathbf{y} := \mathbf{V}_\mathbf{y}^\top\mathbf{x} = \begin{bmatrix} \mathbf{V}_\mathbf{y}'^\top & 0 & 0 \\ 0 & \mathbf{V}_\mathbf{y}'^\top & 0 \\ 0 & 0 & \ddots \end{bmatrix} \mathbf{x}$$

where $\mathbf{V}'_\mathbf{y}$ is the same as in VTB.

Plate (1995) initially proposed the HRR VSA in order to represent complex compositional structures, specifically ones used for language processing and reasoning, with distributed representations like neural networks. Jackendoff proposed four linguistic challenges involving how to neurally represent the compositional structure and rules of language that previously divided linguistic theory and connectionist cognitive neuroscience. VSAs solve these four problems (Gayler, 2004), which supports the idea that VSAs are useful in studying linguistics and reasoning within the context of neural networks. For example, VSAs have been used in neural models to represent lexical relations and recursively structured sentences successfully (Crawford et al., 2016). The quality of structural representation and mathematical ability to query information from these VSA representations naturally lends this method to QA tasks where representing and extracting relational information pertaining to a set of concepts is necessary.

As a simple example, a scene of a dog holding a stick could be represented with VSAs in a slot-filler fashion as: $\mathbf{scene} = \mathbf{subject} \circledast \mathbf{dog} + \mathbf{verb} \circledast \mathbf{holds} + \mathbf{object} \circledast \mathbf{stick}$, given the slot vectors $\mathbf{subject}, \mathbf{verb}, \mathbf{object} \in \mathbb{R}^d$ and filler vectors $\mathbf{dog}, \mathbf{holds}, \mathbf{stick} \in \mathbb{R}^d$. One could then query the subject of the scene with unbinding: $\mathbf{scene} \circledast \mathbf{subject}^+ \approx \mathbf{dog} + noise$, which can be cleaned up to the exact $\mathbf{dog}$ vector by finding the VSA vector in the vocabulary with the highest similarity to the result. We use these techniques for representing structure to capture concept relations in a knowledge graph to propose a novel question answering neural network model.

4 Methods

Given a multiple-choice question q and an answer option a, as in QA-GNN (Yasunaga et al., 2021), the purpose of the model is to score the plausibility of each (q, a) pair from the set of all answer options by performing joint reasoning using a (q, a) context, $\mathbf{z}$, generated from a PLM text encoder, and a working graph G_w that contains relations and concepts pertaining to each specific (q, a) pair. In QA-GNN, the graph reasoning portion of the model consists of a specific type of GNN called a Graph Attention Network (GAT; Velickovic et al. (2018)). As a replacement for the GAT in QAVSA, a single VSA vector representation of the (q, a) graph is generated. This representation is used as input to a simple k-layer MLP to realize the graph reasoning portion of QAVSA. Using this learned MLP along with the PLM (q, a) context embeddings, the (q, a) pair is scored, as shown in Figure 1.

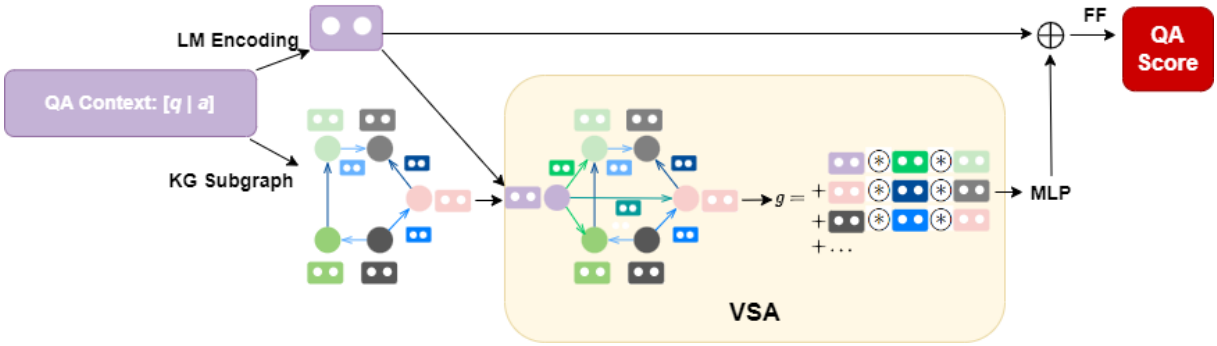

Figure 1: QAVSA model outline. The QA context, $[q|a]$, is inputted to a PLM to generate an LM Encoding for the context and is also used to generate a KG subgraph. The LM encoded QA context is added to the graph, and the graph is converted to a single vector using VSA. The VSA representation is feed through a k-layer MLP, concatenated with the original QA context encoding, and passed through a single FF layer to score the QA pair.

4.1 Graph Data Pre-processing

To generate the working graph G_w, we follow the exact pre-processing technique described in Lin et al. (2019). The external domain-specific or world knowledge relevant to the answer question task is defined by a knowledge graph $G = (V, E)$ made up of a set of nodes, V, and a set of directed edges to capture relations, $E \subseteq V \times R \times V$, connecting the nodes with relation types from the set R.

The nodes of the working subgraph G_w are selected by first linking the concepts from the question, V_q, and from the answer, V_a, to nodes in G, where $V_q \cup V_a = V_{q,a} \subset V$. All of the nodes on a 2-hop path between the nodes in $V_{q,a}$, i.e., all nodes in V related to two of the nodes in $V_{q,a}$ are also included in the working graph, producing V_w. Finally, V_w is pruned down to 200 nodes by scoring the relevance of each node to the (q, a) pair using a PLM as described in Yasunaga et al. (2021). All of the edges connecting each pair of nodes in V_w, defined as $E_w \subset E$, are included in the final working graph, $G_w = (V_w, E_w)$.

Following Yasunaga et al. (2021), the initial 1024-dimensional embeddings of these nodes are defined by feeding each triple composed of a head, relation and tail entity, $(h, r, t) \in V_w \times R \times V_w$, as a sentence into a PLM text encoder. The representations for each concept are pooled using the corresponding portion of each triple that they appear in. Computing relation embeddings r in Yasunaga et al. (2021) in this way was unnecessary, as they represent the relation type as a one-hot vector for their GNN module. However, in our approach, each relation is a distributed representation, r, so we use an initial vector embedding computed in the same way as each graph concept embedding is computed.

Feng et al. (2020) provide embeddings computed for each concept in the graphs used. However, these embeddings do not include relations. As a result, we computed all embeddings following the process defined in Lin et al. (2019).

4.2 Model

Given the working graph for a (q, a) pair, $G_w = (V_w, E_w)$, and initial concept and relation embeddings $\mathbf{v_i}, i \in [0, 1, \ldots, |V_w|]$, $\mathbf{r_j}, j \in [0, 1, \ldots, |R|]$, the VSA representation of a given triple can be computed as follows. For triple $triple_k = (h_k, e_k, t_k), k \in [0, 1, \ldots, |E_w|]$, where h_k, e_k, t_k specifies the type of entity for the head, relation, and tail of the triple, respectively, we bind each of the elements together using the binding operation of the given VSA: $\mathbf{triple_{k_{vsa}}} = \mathbf{v_{h_k}} \circledast \mathbf{r_{e_k}} \circledast \mathbf{v_{t_k}}$. The working graph VSA representation can be calculated by adding up all the triple VSA representations in the graph:

$$\mathbf{g_{vsa}} = \sum_{k=1}^{|E_w|} \mathbf{triple_{k_{vsa}}} = \sum_{k=1}^{|E_w|} \mathbf{v_{h_k}} \circledast \mathbf{r_{e_k}} \circledast \mathbf{v_{t_k}}.$$

Since circular convolution is commutative, $\mathbf{triple_{k_{vsa}}}$ does not contain information on the direction of the relation, so a specific permutation σ can be applied to either the head or tail element of each triple to specify the directionality of the relation. To query this triple for a permuted concept, σ^{-1} is applied after unbinding.

Given a QA input for question q and answer option a, an LM representation of the context is generated with a pretrained encoder to generate $LM(q|a) = \mathbf{z}$. We can also integrate $\mathbf{z}$ into $\mathbf{g_{vsa}}$, analogous to Yasunaga et al. (2021),

by forming new triples that bind $\mathbf{z}$ with two new defined relation SPs, **IsAnswerConcept** and **IsQuestionConcept**, along with the corresponding answer and question concepts in $\mathbf{g_{vsa}}$. These triples are then added to $\mathbf{g_{vsa}}$ like usual.

For example, the question in the CSQA dataset "What is the primary purpose of cars?" has the answer options {cost money, slow down, move people, turn right}, with the correct answer being "move people". The subgraph for the QA context [What is the primary purpose of cars? move people] has question concepts V_q = {PURPOSE, CAR, PRIMARY, CARS}, answer concepts V_a = {PEOPLE, MOVE, MOVE_PEOPLE}, and many intermediate concepts, along with a set of triples, E = {(MOVE, ANTONYM, STOP), (CAR, CAPABLEOF, GO_FAST), ... }. The graph VSA vector is computed as

$$\mathbf{g_{vsa}} = (\mathbf{MOVE} \circledast \mathbf{ANTONYM} \circledast \mathbf{STOP})_{\mathbf{vsa}} + (\mathbf{CAR} \circledast \mathbf{CAPABLEOF} \circledast \mathbf{GO\_FAST})_{\mathbf{vsa}} + \ldots$$

$\mathbf{g_{vsa}}$ is then used as the input to a k-layer MLP with dropout and layer normalization, $MLP(\mathbf{g_{vsa}}) = \mathbf{g^*_{vsa}}$, and is responsible for learning to update the VSA representations within the graph vectors to solve the task (see Figure 1).

A plausibility score, i.e. the probability of answer a being correct, is computed with $p(a|q) \propto \exp(FF(\mathbf{z} \bigoplus \mathbf{g^*_{vsa}}))$, where the initial QA context $\mathbf{z}$ is concatenated with the final graph VSA representation, $\mathbf{g^*_{vsa}}$, and is passed through a final feedforward layer.

During training, the cross entropy between the plausibility scores of all answer options are computed, and are backpropagated through both the LM and VSA MLP components of the model.

5 Experiments

In order to perform a comparison to QA-GNN, we evaluate QAVSA on the same three datasets that were used to evaluate QA-GNN in Yasunaga et al. (2021). However, we hyperparameter tune our model rather than keeping parameters the same as those used in QA-GNN to maximize accuracy on the benchmark development splits.

5.1 Datasets

CommonsenseQA (CSQA) (Talmor et al., 2019) is a 5-way multiple choice commonsense reasoning task that requires different types of commonsense knowledge. The dataset has 12,102 questions, and the inhouse train/dev/test split is adapted from Lin et al. (2019) as 8500/1221/1241.

OpenbookQA (OBQA) (Mihaylov et al., 2018) is a 4-way multiple choice dataset aiming to assess human understanding of a subject in an open-book setting. The dataset consists of a list of 1326 science facts along with 5957 elementary school science questions with a train/dev/test split of 4957/500/500.

MedQA-USMLE (MedQA) (Jin et al., 2021) is a 4-way multiple choice dataset based on questions from the United States Medical License Exams (USMLE). The english version of the dataset has 12,723 questions, with a train/dev/test split of 10178/1272/1273.

For CSQA and OBQA, the external KG used is ConceptNet (Speer et al., 2017). ConceptNet consists of 799,273 common words or phrases connected by edges of 17 different merged relation types, after preprocessing. The method to initialize the concept and relation embeddings for ConceptNet are also described in Section 4.1, and uses PLMs BERT-large or RoBERTa-Large. Following Yasunaga et al. (2021), RoBERTa-Large is he PLM used to encode the QA contexts in QAVSA for CSQA, and AristoRoBERTa (Clark et al., 2020) is used for the QA contexts for OBQA.

The external KG used for MedQA is the Unified Medical Language System (UMLS; Bodenreider (2004)), a popular biomedical knowledge base with 300K nodes, 1M edges, and 98 relation types. The PLM encoder used to generate concept and relation vector embeddings is BioLinkBERT, following Yasunaga et al. (2022a), which is a specific version of LinkBERT that utilizes medical document hyperlinks. BioLinkBERT is pretrained on PubMed with citation links to perform both masked language modeling and document relation prediction (Yasunaga et al., 2022b).

5.2 Implementation and Training Details

Because we had to recompute concept embeddings for ConceptNet and UMLS, we reproduced results from QA-GNN with these new embeddings as a baseline. The LM and MLP learning rates and learning schedule for QA-GNN are kept to their

Model	IH Dev. Acc.	IH Test Acc. (%)
RoBERTa-Large* (w/o KG)	76.27 (±0.45)	70.23 (±0.80)
+QA-GNN*	75.97 (±0.64)	**71.88 (±1.11)**
+QAVSA	**76.61 (±0.54)**	70.56 (±0.72)

Table 1: Accuracy and standard deviation on CSQA inhouse dev. and test splits. Reproduced results (*) use reproduced node embeddings and all results are averaged over 5 different seeds.

Model	Dev. Acc. (%)	Test Acc. (%)
AristoRoBERTa* (w/o KG)	81.32 (±0.61)	81.00 (±0.65)
+QA-GNN*	**81.92 (±0.78)**	80.36 (±1.63)
+QAVSA	81.76 (±1.41)	**81.92 (±0.88)**

Table 2: Test accuracy on OBQA. Reproduced results (*) use reproduced node embeddings and all results are averaged over 5 different seeds.

original values for all three datasets since their model training proved to be unstable with the parameters that were optimized for QAVSA. The number of training epochs for QA-GNN are 15, 40, and 30 for CSQA, OBQA, and MedQA, respectively, to match the original amount epochs, whereas QAVSA is trained for 15 epochs on each dataset. As a baseline to both QA-GNN and QAVSA model results, we reran our model consisting of only the PLM encoder and final layer scoring components, with all other parameters remaining unchanged.

Hyperparameter tuning was done using a Tree-structured Parzen Estimater as a sampler for both OBQA and CSQA. The variables optimized during tuning and final model parameter values are shown in Appendix A.1 in Tables 7 and 6, respectively.

Although the QA-context embedding, $\mathbf{z}$, is added to the graph in Figure 1, the QAVSA results in Section 6.1 are produced from a version of QAVSA that uses working graphs without adding $\mathbf{z}$ to them.

6 Results

6.1 Main Results

As shown in Table 1, QAVSA has an improvement in mean accuracy of 0.34% and 0.61% compared to QA-GNN and RoBERTa-Large, respectively, on the CSQA inhouse dev. split. On the inhouse test split however, QA-GNN outperforms QAVSA by a difference of 1.32% and improves upon the baseline by 1.65%.

On the OBQA dataset, QA-GNN has the best performance on the dev. split with a mean accuracy of 81.92%, as seen in Table 2. QAVSA is close behind with a mean accuracy of 81.76%, which is a 0.44% increase compared to the AristoRoBERTa baseline. However, on the test split QAVSA outperforms QA-GNN by a larger margin with a mean accuracy of 81.92%, which is a 0.92% improvement on the AristoRoBERTa baseline and a 1.56% improvement over QA-GNN.

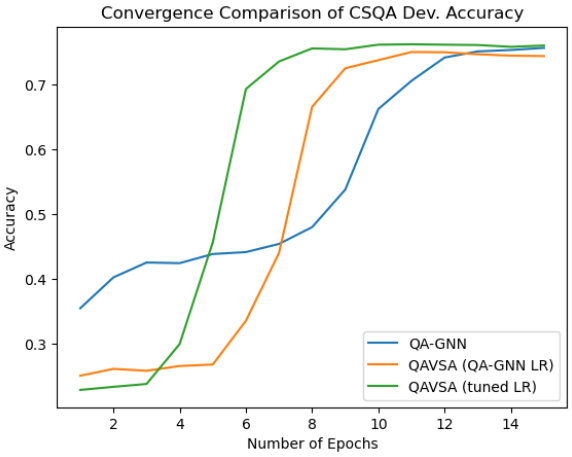

Figure 2: Mean accuracy of QA-GNN, QAVSA using our tuned LR schedule, and QAVSA using the LR schedule of QA-GNN on the CSQA dev. split for each epoch, averaged over 5 seed runs.

As shown in Table 3, QAVSA outperforms QA-GNN and the BioLinkBERT baseline on MedQA by a mean accuracy of 0.37% and 0.61%, respectively, on the dev. split and by 0.42% and 1.16%, respectively, on the test split.

Also, as shown in Figure 2, QAVSA converges faster during training. On average, it takes 11.2, 8.8, and 6.8 epochs to reach within 5% of each run's maximum accuracy for QA-GNN, QAVSA (QA-GNN LR schedule), and QAVSA (our LR sched-

Model	Dev. Acc. (%)	Test Acc. (%)
BioLinkBERT* (w/o KG)	43.55 (±0.08)	43.96 (±0.12)
+QA-GNN*	43.79 (±0.31)	44.7 (±0.47)
+QAVSA	**44.16 (±0.57)**	**45.12 (±0.70)**

Table 3: Test accuracy and standard deviation on MedQA. Reproduced results are denoted with * and all results are averaged over 3 runs.

ule), respectively. Demonstrating that QAVSA can be trained 39% faster than QA-GNN.

Overall, these results suggest that QAVSA performs similarly to QA-GNN, but has a significantly faster convergence time during training.

6.2 Model Variations and Ablation

Results of several QAVSA model variations on the OBQA benchmark are shown in Table 4, with all other architecture parameters in Table 6 kept constant. The QAVSA result in Table 4 corresponds to the results of one of the five seeds from Table 2. Including the QA context into the working graph drops the dev. and test accuracy by $\sim 3\%$, suggesting that for this parameter configuration, dynamically updating the graph representation in this way either muddles the original graph representation, or creates a VSA representation that is more difficult to learn by the neural network.

Introducing directionality in the VSA triples by means of permutation on the head or tail entities in the triple does not improve accuracy, which indicates that binding through circular convolution stores enough semantic information from the graph for the task. Furthermore, the HRR VSA is superior to other non-commutative algebras like VTB and TVTB with this architecture, with improvements of 1.6% and 2.2% in dev. accuracy and 3.2% and 1.4% in test accuracy, compared to VTB and TVTB, respectively.

Applying normalization to the graph VSA vectors only drops performance when the concept vectors are also normalized. This indicates that including some type of information of the magnitude of either the graphs or concepts in the VSA vectors is useful for question answering. Not normalizing both graphs and concepts resulted in graph VSA vectors with too wide of a magnitude range for stable training.

Also, BERT performs fairly similarly to RoBERTa in initializing concept and relation embeddings (Section 4.1), with a drop in accuracy only on the dev. split.

Results for an ablation study on the OBQA benchmark are shown in Table 5. Replacing BERT-encoded relation embeddings with random unit-length 1024-D vectors, with a maximum similarity to the concept vocabulary of 0.3, dropped the test accuracy by 2%. However, the dev. accuracy remained fairly consistent, suggesting that the dissimilarity of the relation vectors may be enough to properly represent the semantics of the graph.

Removing node pruning, so that all the nodes and respective edges are included subgraph rather than just the top 200 nodes, dropped test accuracy the most significantly, which may suggest that the extra nodes included in the graph vectors did not add triples useful to reasoning over the question and answer concepts.

Layer normalization between the VSA MLP layers also seems to be important for learning over these graph representations, as seen in the 1.8% drop in dev. and test accuracy when it is removed.

6.3 Explainable Graph VSA Representations

We can analyze the effectiveness of the MLP portion of QAVSA by computing the similarities of each triple VSA vector in a graph to the initial graph vector, $\mathbf{g_{vsa}}$, and the final graph vector, $\mathbf{g^*_{vsa}} = MLP(\mathbf{g_{vsa}})$ using the similarity operator defined in Section 3. Such similarities can be used to determine which triples become the most prominent in the graph vector through the MLP transformation. In the top 20 most similar triples from the initial graph representation in Section 4.2, we find triples relating to the concept CAR, such as (CAR, RELATEDTO, DRIVE), (MOTOR, RELATEDTO, CAR), and (STOP, RELATEDTO, CAR), but it does not contain any triple relating the answer "move people". After the MLP, however, in the top 20 most similar triples to $\mathbf{g^*_{vsa}}$, the triples (STREET, USEDFOR, TRANSPORTATION), (TRANSPORTATION, RELATEDTO, CAR), and (STREET, RELATEDTO, CARS) appear. Given that the concept TRANSPORTATION is closely related to the answer "move people" and does not appear in the most similar initial triples, we can see that the $\mathbf{g^*_{vsa}}$ attends more to

Model Version	Dev. Acc. (%)	Test Acc. (%)
QAVSA	82.6	**83.4**
+ QA-context	79.8	80.2
+ Permutation (head)	81.8	82.6
+ Permutation (tail)	81.6	83.2
+ Graph norm	81.6	82.4
+ Graph norm - Concept not norm	**83.0**	82.8
RoBERTa Embeddings	81.0	**83.4**
VTB	81	80.2
TVTB	80.4	82.0

Table 4: Accuracy of different model variations on dev. and test splits of OBQA.

Model Version	Dev. Acc. (%)	Test Acc. (%)
QAVSA	**82.6**	**83.4**
− BERT rels	82.4	81.4
− Node Pruning	82	80.4
− Layer Norm	80.8	81.6

Table 5: Accuracy on OBQA of ablation study. The minus sign (−) indicates what was removed.

concepts and reasoning paths necessary for answering the question. Since the learned graph transformations are not perfect, some unrelated triples do appear in $\mathbf{g}^*_{\mathbf{vsa}}$, such as (CHROME, RELATEDTO, METAL) and (FERRY, RELATEDTO, MOVE).

This method of analysis can also explain incorrect answers produced by the model. For the question from the CSQA dev. split, "What do audiences clap for?", QAVSA predicted the answer to be "hockey game" rather than the correct label "show". Looking at the 20 most similar triples in $\mathbf{g}^*_{\mathbf{vsa}}$ for the answer option "show", there are triples relating to AUDIENCE and SHOW, like (AUDIENCE, RELATEDTO, THEATRE) and (PROGRAM, RELATEDTO, SHOW), but there are no triples related to CLAP. Looking at the 20 most similar triples to $\mathbf{g}^*_{\mathbf{vsa}}$ for the answer option "hockey game", a reasoning path can be drawn with (SPORT, RELATEDTO, PLAY), (PLAY, RELATEDTO, EVENT), (BEAT, RELATEDTO, EVENT), and (CLAP, HAS_SUB_EVENT, BEAT). Two different definitions of BEAT are used in the prior triples, making the reasoning path illogical. The concept "CLAP" appears in 3 out of the top 20 triples for answer option, thus potentially leading QAVSA to choose this option as the most correct answer. With this particular example, this VSA-style analysis suggests that QAVSA may attend to multiple meanings of the same concept incorrectly, which could be useful information for finetuning the method further or applying it to other tasks.

7 Discussion

On the CSQA and OBQA datasets, there is no definitive top performer between QA-GNN and QAVSA due to the fact that both models have the best performance on one of the dataset splits. The accuracy of QA-GNN is slightly lower than PLM baseline for the CSQA inhouse dev. split and the OBQA test split, and this is most likely due to a combination of larger variance of model accuracy between seeds along with the fact that the model learning rate was not tuned for our recomputed concept embeddings.

For MedQA however, QAVSA slightly outperforms QA-GNN on both the dev. and test splits. MedQA is a significantly harder than CSQA and OBQA due to the nature of the in-depth medical questions asked. This is evidenced by the difference in accuracy of more than 30% compared to CSQA and OBQA. This more consistent performance increase may suggest that the increased semantic complexity of the concepts and relations in UMLS benefit more from the structured VSA representations generated and reasoned over in QAVSA rather than using multi-relational GNNs to update concept embeddings.

Although QAVSA does not consistently outperform QA-GNN, the architecture is much simpler than the Graph Attention Network in QA-GNN as it feeds VSA vectors into standard MLP layers. There is no requirement to use node embeddings matrices during computation along with linear and nonlinear transformations on node and relation type embeddings to perform message passing between concepts. Also, there is no requirement to use graph attention layers to create attention weights

on the relations between concepts. In QAVSA, the attention on the relational edges within the graph arises naturally and can be analyzed as shown in Section 6.3.

Additionally, the QAVSA memory requirements for its graph representation is constant at the dimensionality of the VSA vector, d. This compares favorable to GNNs that require an $N \times d$ node embedding matrix and an $N \times N$ adjacency matrix. For these benchmarks, QA-GNN requires graphs with exactly 200 nodes for each QA pair. If one wanted to scale up the number of nodes in the graph significantly, the memory resources required would grow quadratically. In contrast, with an increase in the number of graph nodes and edges, the memory requirements for QAVSA are constant.

8 Conclusion

We presented QAVSA, a new type of model that leverages VSA-represented knowledge graphs along with general linguistic knowledge from PLMs to perform reasoning on MCQA benchmarks. Through a direct comparison to the GNN-based model QA-GNN, we exhibit the ability of QAVSA to perform similarly to QA-GNN on three datasets, while using a simpler k-layer MLP reasoning module. We also demonstrate faster convergence during training than QA-GNN and highlight the explainability of our model outputs through our VSA graph representations.

For future study, our method of representing knowledge graphs with VSAs could be useful in a wide variety of knowledge graph QA tasks involving information retrieval, like multi-hop reasoning (Lan et al., 2021). There are many other ways that KGs are integrated into LLMs, such as using them to augment LLM input or using them as training objectives during LLM pretraining, so having an efficient VSA representation of these KGs may be beneficial to these methods (Pan et al., 2024). GNNs are also widespread for tasks outside of natural language processing, such object detection, chemical reaction prediction, and disease classification, and it is worthwhile to determine if our VSA-based approach is useful for representing structures that are not linguistic (Zhou et al., 2020).

Limitations

Our method depends on already constructed knowledge graphs (i.e., ConceptNet, UMLS), and specifically with these benchmarks, a predefined subgraph generation process. Thus, the quality of our graph VSA representations for question answering tasks is dependent on the quality of the initial graph construction.

Similarly, the quality of the intitial concept and relation embeddings is of great importance. If the concept vector embeddings are too low-dimensional, have large variance in their magnitude, or are too similar to each other, the individual triple or graph VSA representations may not be able to contain many graph triples.

Ethical Concerns

Our proposed model uses pretrained language models, and because of this, any biases or stereotypes in their training data may be reflected in model outputs.

Broader Impact

The augmentation of PLMs with GNNs is widespread, so many further studies could be conducted to compare this VSA-based method to any of these models. Also, this paper encourages the exploration of augmenting large language models using VSAs outside the context of KGs and GNNs.

Acknowledgements

The authors would like to thank the members of the Computational Neuroscience Research Group (CNRG) for discussions that helped improve this paper. This work was supported by CFI (52479-10006) and OIT (35768) infrastructure funding as well as the Canada Research Chairs program, and NSERC Discovery grant 261453.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu

Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. PaLM 2 Technical Report. ArXiv:2305.10403 [cs].

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(Database issue):D267–D270.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, and Michael Schmitz. 2020. From F to A on the New York Regents Science Exams — An Overview of the Aristo Project. *AI Magazine*, 41(4):39–53.

Eric Crawford, Matthew Gingerich, and Chris Eliasmith. 2016. Biologically plausible, human-scale knowledge representation. *Cognitive science*, 40(4):782–821.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Ross W. Gayler. 2004. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. *ArXiv*, abs/cs/0412059.

Jan Gosmann and Chris Eliasmith. 2019. Vector-Derived Transformation Binding: An Improved Binding Operation for Deep Symbol-Like Processing in Neural Networks. *Neural Computation*, 31(5):849–869.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023a. Large Language Models Can Self-Improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Yongfeng Huang, Yanyang Li, Yichong Xu, Lin Zhang, Ruyi Gan, Jiaxing Zhang, and Liwei Wang. 2023b. MVP-Tuning: Multi-View Knowledge Retrieval with Prompt Tuning for Commonsense Reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13417–13432, Toronto, Canada. Association for Computational Linguistics.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What Disease Does This Patient Have? A Large-Scale Open Domain Question Answering Dataset from Medical Exams. *Applied Sciences*, 11(14):6421. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing Format Boundaries with a Single QA System. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

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 *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4483–4491, Montreal, Canada. International Joint Conferences on Artificial Intelligence Organization.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. UNICORN on RAINBOW: A Universal Commonsense Reasoning Model on a New Multitask Benchmark. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13480–13488. Number: 15.

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. ArXiv:1809.02789 [cs].

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

T.A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641. Conference Name: IEEE Transactions on Neural Networks.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral, Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. 2023. Towards Expert-Level Medical Question Answering with Large Language Models. ArXiv:2305.09617 [cs].

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1). Number: 1.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. ArXiv:1811.00937 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Kuan Wang, Yuyu Zhang, Diyi Yang, Le Song, and Tao Qin. 2022. GNN is a Counter? Revisiting GNN for Question Answering. In *International Conference on Learning Representations*.

Yichong Xu, Chenguang Zhu, Shuohang Wang, Siqi Sun, Hao Cheng, Xiaodong Liu, Jianfeng Gao, Pengcheng He, Michael Zeng, and Xuedong Huang. 2022. Human Parity on CommonsenseQA: Augmenting Self-Attention with External Attention. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, volume 3, pages 2762–2768. ISSN: 1045-0823.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2021. Fusing Context Into Knowledge Graph for Commonsense Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1201–1207, Online. Association for Computational Linguistics.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy S. Liang, and Jure Leskovec. 2022a. Deep Bidirectional Language-Knowledge Graph Pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022b. LinkBERT: Pretraining Language Models with Document Links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, Dublin, Ireland. Association for Computational Linguistics.

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 *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Zi Ye, Yogan Jaya Kumar, Goh Ong Sing, Fengyan Song, and Junsong Wang. 2022. A Comprehensive Survey of Graph Neural Networks for Knowledge Graphs. *IEEE Access*, 10:75729–75741. Conference Name: IEEE Access.

Zhangdie Yuan, Songbo Hu, Ivan Vulić, Anna Korhonen, and Zaiqiao Meng. 2023. Can Pretrained Language Models (Yet) Reason Deductively? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1447–1462, Dubrovnik, Croatia. Association for Computational Linguistics.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2021. GreaseLM: Graph REASoning Enhanced Language Models. In *International Conference on Learning Representations*.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

A Appendix

A.1 Model Parameters

Final model and training parameters are shown in Table 6. The parameters optimized for OBQA are also used for MedQA. k specifies how many layers the MLP will have. LR schedule cycles defines how many cosine periods are in the LR schedule. The encoder learning rate (LR) specifies the LR for whatever PLM is in use to encode the QA context, and the decoder LR applies to all other components of the model. The unfreeze epoch defines after how many epochs do the PLM weights unfreeze. The ranges for these variables during hyperparameter tuning are shown in Table 7.

Variable	CSQA Value	OBQA Value	MedQA Value
k	5	4	4
Epochs	15	15	15
LR Schedule	cosine w/ restarts	cosine w/ restarts	cosine w/ restarts
LR Schedule Cycles	1	2	1
Warmup Steps	200	200	200
Batch Size	64	64	64
Mini Batch Size	8	8	2
Encoder LR	1.77e-5	4.17e-5	4.17e-5
Decoder LR	3.71e-2	3.41e-2	3.41e-2
Unfreeze epoch	3	3	3
Dropout (VSA MLP)	0.2	0.4	0.4
Dropout (final layer)	0.4	0.8	0.8

Table 6: Model parameter values on all experiment datasets.

Variable	Range
k	{3, 4, 5}
LR Schedule Cycles	{1, 2, 3}
Encoder LR	[1e-7, 5e-4]
Decoder LR	[1e-5, 5e-2]
Unfreeze epoch	{0, 3, 6}
Dropout (VSA MLP)	{0.2, 0.4, 0.6, 0.8}
Dropout (final layer)	{0.2, 0.4, 0.6, 0.8}

Table 7: Variable ranges for hyperparameter tuning for CSQA and OBQA.