The APVA-TURBO Approach To Question Answering in Knowledge Base

Yue Wang[†], Richong Zhang[†*], Cheng Xu[†] and Yongyi Mao[‡]
[†]BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University, China
[‡]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
[†]{wangyue16, zhangrc, xucheng}@act.buaa.edu.cn
[‡]yymao@site.uottawa.ca

Abstract

In this paper, we study the problem of question answering over knowledge base. We identify that the primary bottleneck in this problem is the difficulty in accurately predicting the relations connecting the subject entity to the object entities. We advocate a new model architecture, APVA, which includes a verification mechanism responsible for checking the correctness of predicted relations. The APVA framework naturally supports a well-principled iterative training procedure, which we call turbo training. We demonstrate via experiments that the APVA-TUBRO approach drastically improves the question answering performance.

Title and Abstract in Chinese

面向知识库问答的 APVA-TURBO 方法

本文主要围绕目前基于知识库的问答方法中的存在的问题展开研究。经过对现有各类问答方法的调查与分析，我们发现目前知识库问答的瓶颈主要体现在关系预测上，即如何准确地预测出问题中的实体与答案实体之间的关联，是需要面临的最大挑战。因此本文在现有问答框架的基础上，加入了负责对预测关系的可靠性进行评价检验的验证机制，用以增强关系预测效果，并提出了一种新的知识库问答框架“APVA”。另外，文中为 APVA 设计了一种具有良好理论基础的迭代训练过程，我们称之为“涡轮式(turbo)训练”。通过实验证明，APVA-TUBRO 方法可以在问答数据集上取得优异效果，大大提升了目前问答方法的准确性。

1 Introduction

Knowledge bases (KBs), such as YAGO, DBPedia, and Freebase contain an enormous volume of facts, or knowledge, about the real world. Each of these facts is represented as a triple (s, r, o), where s is a *subject entity*, r is a *relation*, and o is an *object entity*. This representation allows the KB to be interpreted as a graph in which entities are vertices, triples are edges, and relations are edge labels. Such a structured representation has enabled the development of query languages, such as RQL(Karvounarakis et al., 2002), Versa(Ogbuji, 2005) and SPARQL(Prud et al., 2006), for retrieving knowledge stored in the KB. Despite the usefulness of these languages, it is highly desirable that the query-processing engine of a KB is capable of processing queries presented in human language directly. This stimulates intense research in question answering over KB.

Briefly, question answering over KB, or QA-KB, aims at taking as input a question presented in a natural language and outputs the answer in terms of a fact triple or a collection of fact triples in the KB. In a general setting, the answer is a set of *paths*, where each path connects the subject entity of the question to the object entity via a sequence of relations. We refer to this relation sequence as a *path label*.

There are two main lines of solutions to QA-KB. The first centres around semantic parsing, which translates the natural-language question into a logical form and uses it to query the KB. This line of

*Corresponding Author: Richong Zhang

Proceedings of the 27th International Conference on Computational Linguistics, pages 1998–2009
Santa Fe, New Mexico, USA, August 20-26, 2018.

solutions includes, for example, (Berant et al., 2013; Dong et al., 2017; Yih et al., 2015; Xu et al., 2016; Reddy et al., 2017).

The second line of solutions centres around learning with neural network models, which follows what we call an APA architecture or its variants. Specifically, the APA architecture includes three components: an *entity alignment* component, which identifies the subject entity implied in the question, a *path label prediction* component, which predicts a path label, and an *object answering* component, which finds the object entitie(s). This line of solutions includes, for example, (Dong et al., 2015; Hsiao et al., 2017; Yin et al., 2016; Yu et al., 2017; Lukovnikov et al., 2017; Jain, 2016; Hao et al., 2017; Bordes et al., 2014).

Despite the success of the existing approaches to QA-KB, particularly that of the APA architecture, we observe that the primary bottleneck for further improving the model performance lies in the model's limited accuracy in label path prediction. Motivated by this observation, this work extends the APA methodology into an "APVA" framework, which incorporates a verification mechanism in the model. Such a mechanism is responsible for checking if the predicted relation or path label is correct and hence improving the answering performance. Unifying APVA's components in a probabilistic modelling set-up, we show that such an APVA architecture allows the joint training of its model components. More specifically, the joint training can exploit coordinate descent in the optimization of the loss function. This naturally gives rise to an iterative training procedure, which we call turbo training. We develop a concrete model and its training algorithm based on this APVA-TURBO approach. Experimental study demonstrates that the proposed APVA-TURBO model outperforms the existing models by a large margin, thereby establishing itself as a new state of the art in QA-KB.

2 Problem Statement

For a given KB $\mathcal{K}$, we will denote its set of entities by $\mathcal{E}$ and its set of relations by $\mathcal{R}$. The KB $\mathcal{K}$ contains a set of factual triples $\mathcal{F}$ in which each triple is in the form of (s, r, o), with $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. For example, (CarlosGomez, baseball.baseballPlayer.position, CenterFielder) is one such triple taken from Freebase. It specifies the fact that Carlos Gomez plays the Center Fielder position in baseball. It is customary to interpret the triple set $\mathcal{F}$ in $\mathcal{K}$ as an edge-labelled graph in which vertices are entities in $\mathcal{K}$, edges are triples in $\mathcal{F}$, and on each edge (s, r, o), r is the edge label.

Additionally the KB $\mathcal{K}$ usually also prescribes a set $\mathcal{T}$ of entity types. Specifically, for each entity $e \in \mathcal{E}$, its type $t(e)$ is a token in $\mathcal{T}$. Note that in KBs like Freebase, an entity e may be associated with multiple types. In this work, for simplicity, we take $t(e)$ as the *primary type* of e (which is referred to as the "notable type" of e in Freebase).

We will denote by q the question presented in natural language. Specifically q is a sequence $(w_1, w_2, \ldots, w_n)$ of words, where each w_i is a word in a vocabulary $\mathcal{V}$, and n depends on q. The objective of the *question answering over KB*, or simply QA-KB, is then to develop an answering algorithm which returns the correct answer triple (s, r, o) for any given question q. For example, when a user asks a question "What position does Carlos Gomez play", the answering algorithm is expected to retrieve the triple (CarlosGomez, baseball.baseballPlayer.position, CenterFielder) from the KB.

The above form of QA-KB is in fact in its most basic form. The problem can be made more general in at least the following two ways. First, the answer to q may be multiple triples which form a path in the KB graph. For example, the answer could involve two triples (a, r_1, b) and (b, r_2, c) concatenated in tandem. This corresponds to the path (a, r_1, b, r_2, c) in the KB graph. We will call such an answer a *multi-hop answer*, and call the pair (r_1, r_2) of edge labels the *path label*. Furthermore, there can be multiple answers for a given question q. For example, if the question is "what are the provinces of Canada", then more than one answer is expected.

This paper in fact deals with a general setting of QA-KB, namely, a question may have *multiple multi-hop* answers. We do impose a restriction that we only consider the case in which all answers of any given question have *the same* path label. We note that this simplification does not imply that our proposed framework and model do not extend further. It is solely due to that this restriction sufficiently well describes the datasets we work with.

Denote by $\mathcal{R}^*$ the set of all finite-length sequences consisting of relations in $\mathcal{R}$. We still denote an

answer, multi-hop or single-hop, as a triple (s, r, o) while considering $r \in \mathcal{R}^*$. For any given question q, let $\mathcal{A}(q)$ be the set of all answers, for q, where each answer is in the form of (s, r, o) with $r \in \mathcal{R}^*$.

A machine learning model for QA-KB relies on a training set $\mathcal{D}$ consisting of many pairs $(q, \mathcal{A}(q))$. We will use $\mathcal{D}_{\text{q}}$ to denote the set of all questions in $\mathcal{D}$. After the model is trained on $\mathcal{D}$, one expects it to output the answers $\mathcal{A}(q)$ for an arbitrary question q, unseen in $\mathcal{D}_{\text{q}}$.

3 APVA Modelling Framework

3.1 The Conventional APA Architecture

Conventionally, a QA-KB model, if put into a probabilistic framework, can be regarded as learning a conditional distribution $p(s, r, o|q)$, where for each configuration of (s, r, o, q), the value $p(s, r, o|q)$ is probability that the triple (s, r, o) is an answer in $\mathcal{A}(q)$. Note that in this formulation, we have suppressed the dependency of (s, r, o) on the given KB $\mathcal{K}$.

By the chain rule of probability,

$$p(s, r, o|q) = p(s|q)p(r|q, s)p(o|q, s, r).$$

A QA-KB model factorizing this way essentially assumes three components: an *entity alignment* component $p(s|q)$ to identify the subject entity s in the question q, a *path label prediction* component $p(r|q, s)$ to predict the path label r that leads the subject entity to the object entities, and an *object answering* component to provide the list of object entities o's each forming an answer (s, r, o). We call such a model an *APA* model.

The existing neural network models for QA-KB largely fall in the APA architecture, or its variants, in which instead of having the three components separable, the model may combine some of the components or have parameters shared between them. We now summarize these models.

In (Yin et al., 2016; Lukovnikov et al., 2017), a word/character-level encoder is used for question encoding and to match the subject entity and the relation in a fact candidate with the question. In (Yu et al., 2017; Hsiao et al., 2017) a bi-directional LSTM is used to score and rank the question and relation pairs. In (Ture and Jojic, 2017), only simple RNN is used both for entity alignment and for relation prediction.The approach of (Jain, 2016) extracts candidate triples and finds a path to the object entity using multi-hop reasoning and refinement. In (Dong et al., 2015; Hao et al., 2017), a multi-column CNN or a cross-attention mechanism is introduced to match the question with the answer path, the answer context and the answer type. In (Bordes et al., 2014), questions and their answer subgraphs from KB are both embedded into a lower dimensional to be matched.

3.2 The APVA Framework

Despite the successes with the APA approaches, we observe that the bottleneck in the existing models lies primarily in the accuracy of predicting the path labels. In fact our experiments suggest that, if the correct (s, r) can be obtained, a good object answering accuracy can be easily achieved. For example, on the WebQuestion dataset(Berant et al., 2013), we see that a simple SVM classifier followed by a search of the KB (see section 5.2) achieves an answering precision of 89.68% and a recall of 93.20%.

The fact that path label prediction forms the primary bottleneck can be attributed to the enormous number of entities and the extremely high dimension of the question space. Relative to the space of (q, s), the dataset $\mathcal{D}$ is in fact very sparse. This makes path label prediction prone to error and limits the performance of a QA-KB model.

To increase the accuracy of path label prediction, this paper augments the APA models with an additional component, which we call *subject-path verification*. The basic idea here is to use a verification mechanism to check if a predicted pair (s, r) is correct. We call this architecture the *APVA* architecture.

3.2.1 The APVA Architecture

In APVA framework, for each question q, we assume that there is another variable y, taking values in $\{0, 1\}$ and indicating whether the obtained (s, r) is correct (1 for correct). The overall objective of

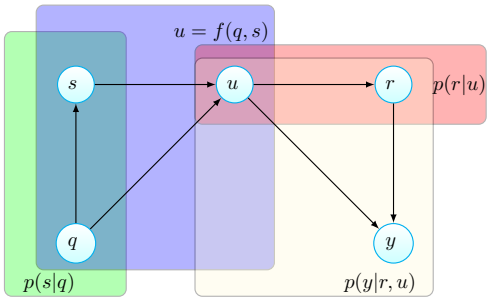


Figure 1: The $p(s, r, y|q)$ part of APVA

learning is then to learn a distribution $p(s, r, o, y|q)$, which factors as

$$p(s, r, o, y|q) = p(s, r, y|q)p(o|q, s, r, y). \tag{1}$$

The term $p(o|q, s, r, y)$ in (1) essentially characterizes the *object answering* component (where we are only interested in the case of $y = 1$). The term $p(s, r, y|q)$ further factors as

$$p(s, r, y|q) = p(s|q)p(r|q, s)p(y|q, s, r) \tag{2}$$

In (2), the terms $p(s|q)$ and $p(r|q, s)$ respectively correspond to an *entity alignment* component and a *path label prediction* component. The term $p(y|q, s, r)$ on the other hand is the new *subject-path verification* component.

For the path label prediction component, APVA further expresses $p(r|q, s)$ by

$$u := f(q, s) \tag{3}$$
$$p(r|q, s) := p(r|u) \tag{4}$$

where f is an appropriate *encoder* (ENC) function, and (4) is referred to as the *decoder* (DEC). Here f serves to compress the high-dimensional representation of (q, s) to a low dimensional space so as to deal with the sparsity of (q, s).

Furthermore APVA expresses $p(y|q, s, r)$ as

$$p(y|q, s, r) := p(y|u, r). \tag{5}$$

Since (5) essentially defines a binary classifier, we call it the *verification classifier* (CLS).

In summary, the APVA framework consists of the following components: entity alignment, path label prediction (ENC-DEC), subject-path verification (CLS) and object answering. Figure 1 shows the overall model architecture of APVA, with the object answering component excluded.

3.2.2 Loss Functions and Training

For each of the components, we can derive an appropriate loss function: the (entity) alignment loss ℓ_{al}, the (path label) prediction loss ℓ_{pr}, the (subject-path) verification loss ℓ_{ve}, and the (object) answering loss ℓ_{an}. It is worth noting that to properly define the verification loss ℓ_{ve}, we need to introduce "negative examples" for CLS.To that end, denote

$$\mathcal{D}_{\text{qsr}} := \{(q, s, r) : q \in \mathcal{D}_{\text{q}}, (s, r, o) \in \mathcal{A}(q)\}$$

That is, $\mathcal{D}_{\text{qsr}}$ is the set of all training examples for path label prediction (ENC-DEC). It is also the set of all positive training examples for the verification classifier CLS.

For each positive example $(q, s, r) \in \mathcal{D}_{\text{qsr}}$, we can create several negative examples by replacing the path label r in (q, s, r) with a random wrong path label r'. This gives rise to a negative training set $\mathcal{D}^{-}_{\text{qsr}}$.

If the four loss functions do not share parameters, then one can separately minimize each loss and learn its parameters. If there is a sharing of parameters among some of these losses , one can take a

joint approach to minimize the (possibly weighted) sum of these losses. A generic approach for this joint minimization is coordinate descent, namely, we hold some parameter fixed and optimize the sum loss over other parameters, and keep iterating this process. We will give a concrete example of this training method in a later section.

Although APVA is derived from probabilistic modelling, its overall methodology extends to non-probabilistic models. Indeed, all we need are those well-defined losses, whether or not they result from probabilistic modelling.

4 The APVA-TURBO Model

We now describe a concrete model in the proposed APVA framework, which we refer to as the APVA-TURBO model. The model can be divided into three models which are trained separately. They are the Entity Alignment Model, the Prediction-Verification Model, and the Object Answering Model. Note that the Prediction-Verification Model is a joint model containing both path label prediction and subject-path verification.

The key innovation in this model architecture is the Prediction-Verification Model, where the Entity Alignment Model is a modest extension of a previous model and the Object Answering Model is a standard application of SVM on a selected choice of feature. To stay focused, details of the Entity Alignment Model and Object Answering Model are presented in section 5.2.

We now present the central piece of APVA-TURBO, the Prediction-Verification Model. It decomposes further into an encoder-decoder (or ENC-DEC) model for *path label prediction* and a classifier model (CLS) for subject-path verification.

4.1 ENC-DEC Model

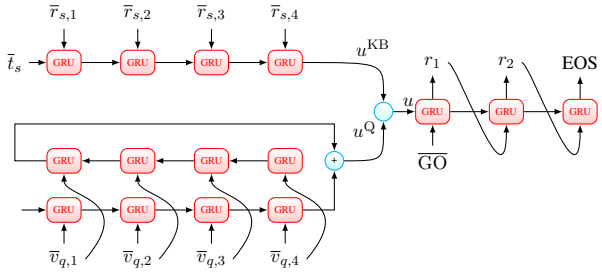


Figure 2: ENC-DEC Model

Recall that in APVA, the ENC component, Equation (3), encodes (q, s) into a vector u and the DEC component, Equation (4), maps u to a desired distribution on the path labels in $\mathcal{R}^*$. The structure of ENC-DEC in APVA-TURBO is shown in Figure 2, which we describe below.

For the identified subject entity s, let $\mathcal{R}(s) := (r_{s,1}, r_{s,2}, \ldots, r_{s,L})$ be the list of all relations in $\mathcal{R}$ that are labels of at least one edge in the KB graph connecting to entity s. Let $\overline{\mathcal{R}}(s) := (\overline{r}_{s,1}, \overline{r}_{s,2}, \ldots \overline{r}_{s,L})$ be the embeddings of $\mathcal{R}(s)$ and $\overline{t}_s$ be the embedding of the type $t(s)$ of s , both of which are obtained from a KB schema embedding model, called KSE. KSE is an separately pre-trained model to embed the relations and types in KB in an Euclidean space, by using an adaptation of the TransH (Wang et al., 2014) to the "schema graph", which is a graphical representation of the KB schema set $\mathcal{S}$ of all "schema triples" $(t, r, t') \in \mathcal{T} \times \mathcal{R} \times \mathcal{T}$ for which knowledge triples (s, r, o) with $t(s) = t$ and $t(o) = t'$ are allowed. We take $\overline{t}_s$ as the initial state of a GRU network, and take $\overline{\mathcal{R}}(s)$ as the input sequence to the network. We denote the final state of the GRU network by u^{KB}. We note that u^{KB} encodes the subject entity s and its relative position in $\mathcal{K}$. We refer to u^{KB} as the *KB-based encoding*. In this encoding step, although order of the elements in $\overline{\mathcal{R}}(s)$ is irrelevant, we still feed them to the GRU network sequentially due to its superiority in processing variable-length input and automatically selecting relevant feature.

On the other hand, for the question $q = (w_1, w_2, \ldots, w_n)$, its subsequence $\mu := (w_m, w_{m+1}, \ldots, w_{m+k})$ of words that corresponds to the subject entity s in the KB $\mathcal{K}$ is referred to

as the *subject mention*. we remove the subject mention μ from the word sequence and replace it with a placeholder token "$\langle e \rangle$" representing the subject entity s. Let $v := (v_{q,1}, v_{q,2}, \ldots, v_{q,n'})$ denote the modified word sequence from q, and let $\overline{v} := (\overline{v}_{q,1}, \overline{v}_{q,2}, \ldots, \overline{v}_{q,n'})$ be its word2vec(Mikolov et al., 2013) embeddings. We simply create a random embedding for "$\langle e \rangle$" and include it in the dictionary.

The sequence $\overline{v}$ is then entered to a Bi-GRU network and the sum of the two final states from the network is denoted by u^{Q}. We then set the output u of the ENC model as the concatenation of u^{Q} and u^{KB}. This completes the ENC block.

The DEC block is a simple GRU network following the usual Seq2Seq(Sutskever et al., 2014) decoder structure. The network takes vector u as its initial state. The input to the network at $t = 1$ is the embedding of a contrived special relation label "GO", signalling the beginning of the input. The input at any other time $t > 1$ is the embedding of the $(t-1)^{\mathrm{th}}$ relation r_{t-1} in the path label r. The embedding dictionary of the relations here is a learnable parameter. Each output of the GRU network at time t is passed to a soft-max function (with learnable parameters) to generate the predictive distribution for the t^{th} relation r_t in r.

Under the maximum-likelihood principle, the loss function $\ell_{\mathrm{pr}}(q, s, r)$ of ENC-DEC is the sum of cross-entropy losses, each between a generated distribution at t and a observed relation r_t in r. We note that, as is standard, a special "EOS" token is included in r to indicate the termination of r.

4.2 CLS Model

The verification classifier CLS is constructed as a Multilayer Perceptron (MLP). The input of the MLP is the concatenation of u with the KSE embedding of a relation r. The MLP outputs the predictive distribution of y. Its loss function $\ell_{\mathrm{ve}}(q, s, r, y)$ for training CLS is defined as the cross entropy between the predictive distribution and the observed class label y.

4.3 Turbo Training of ENC-DEC and CLS

Let θ_{enc}, θ_{dec}, and θ_{cls} denote respectively all parameters in ENC, in DEC and in CLS. The total prediction loss is then

$$\mathcal{L}_{\mathrm{pr}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) := \sum_{(q,s,r) \in \mathcal{D}_{\mathrm{QSR}}} \ell_{\mathrm{pr}}(q, s, r; \theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) \tag{6}$$

and the total verification loss is

$$\begin{aligned} \mathcal{L}_{\mathrm{ve}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) := & \sum_{(q,s,r) \in \mathcal{D}_{\mathrm{QSR}}} \ell_{\mathrm{ve}}(q, s, r, 1; \theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) \\ & + \sum_{(q,s,r) \in \mathcal{D}^{-}_{\mathrm{QSR}}} \ell_{\mathrm{ve}}(q, s, r, 0; \theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) \end{aligned} \tag{7}$$

It is worth noting that the loss ℓ_{ve} does not depend on θ_{dec}, since r is observed in the CLS model. It however depends on θ_{enc} through u.

We define the overall loss for training the Prediction-Verification Model as

$$\mathcal{L} := \mathcal{L}_{\mathrm{pr}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) + \lambda \mathcal{L}_{\mathrm{ve}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}). \tag{8}$$

Here λ is chosen larger than 1 by a decent margin. This is because when using the trained model for answering questions, we will primarily rely on CLS rather than ENC-DEC.

We now present a training algorithm, which we call "turbo training", to minimize the loss function $\mathcal{L}$. Turbo training consists of two phases.

Base Phase.Train ENC-DEC, namely minimizing $\mathcal{L}_{\mathrm{pr}}$ over $(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}})$.

Turbo Phase. In this phase, we iterate over the following three steps in order.

A.Train CLS, namely, minimizing $\mathcal{L}_{\mathrm{ve}}$ over θ_{cls} for the current setting of $(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}})$.

B.Train ENC, namely, minimizing $\mathcal{L}_{\mathrm{pr}} + \lambda \mathcal{L}_{\mathrm{ve}}$ over θ_{enc} for the current setting of $(\theta_{\mathrm{dec}}, \theta_{\mathrm{cls}})$.

C.Train DEC, namely, minimizing $\mathcal{L}_{\mathrm{pr}}$ over θ_{dec} for the current setting of $(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}})$.

A diagram showing the turbo training algorithm is given in Figure 3. It can be proved that the three

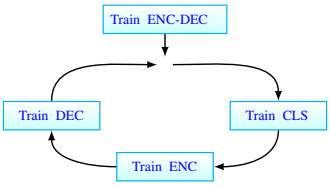


Figure 3: Turbo training

steps in the turbo phase correspond precisely to coordinate-descent minimization of $\mathcal{L}$. Therefore iterations in the turbo phase *provably* decreases the loss $\mathcal{L}$ monotonically and are guaranteed to converge.

In the implementation of turbo training, for each phase and the each step within, the minimization can be carried out via mini-batched SGD over a number of mini-batches. Additionally, when the learning rate of SGD is small, the updates of θ_{enc} in the turbo phase (step B) are primarily dominated by gradient signal of $\mathcal{L}_{\text{ve}}$, due to the large value of λ. Thus step B of the turbo phase can be simplified to minimizing only the loss $\mathcal{L}_{\text{ve}}$.

4.4 Question Answering with APVA-TURBO

At this end, we have completed explaining APVA-TURBO and the training of each component within. After it is trained, APVA-TURBO can be used to solve a QA-KB task according to the following steps.

To answer a question q, the Entity Alignment model is first applied to q and the KB $\mathcal{K}$. A list of M_{s} subject entities are generated as candidates according to their ranking under the model.

Each candidate entity is used as a hypothetic subject entity s, and passed to ENC-DEC together with q and $\mathcal{K}$. In this step, a list of M_{r} candidate path labels are generated. Specifically these candidate path labels are the highest ranked M_{r} path labels among all path labels r for which there is a path with label r in the KB that leaves from s. The ranking of the path labels is according to the learned prediction loss ℓ_{pr}. At the end of this step, a total of $M_{\text{s}}M_{\text{r}}$ (s, r) pairs are generated for question q.

Each of these subject-path pairs (s, r) is then passed to CLS, and the highest ranked (s, r) pair is declared as the correct subject-path pair. Here the ranking is according to the verification loss ℓ_{ve}.

Finally, submitting the declared (s, r) (together with q if needed) to the Object Answering model gives a list $\widehat{\mathcal{A}}(q)$ of answer paths, which APVA-TURBO declares as the answers to question q.

5 Experiments

5.1 Datasets and Pre-Processing

Two popular datasets, SimpleQuestions (Bordes et al., 2015) and WebQuestions (Berant et al., 2013) are used in our experiments.

SimpleQuestions (SQ) consists of 108,442 questions written by human English-speaking annotators. The dataset is partitioned randomly into 75,910 training questions, 10,845 validation questions and 21,687 test questions. The questions in SQ are all single-hop single-answer.

WebQuestions (WQ) consists of 5,810 questions. The dataset is partitioned randomly into 3000 training questions, 778 validation questions and 2032 test questions. The questions in WQ can be multi-hop multi-answer.

The answer triples or paths in SQ and WQ are all in Freebase. A subset of Freebase, FB2M (Bordes et al., 2015), is also used in our study. It includes 2,150,604 entities, 6701 relations and 14,180,937 triples. Since FB2M does not contain the entity type information, we retrieve this information from Freebase to annotate the type for each entity. Specifically, Freebase contains two kinds of types, "common.topic.notable_types" (or "notable type") and "type.object.type". Each entity in Freebase is designed to have a unique notable type while allowed to have many object types. We use notable types as the entity types. For an entity with the notable type missing in Freebase, we annotate it as "NO TYPE".

The answer triples in SQ can all be found in FB2M, whereas not all answer paths in WQ are within FB2M. Thus, for the QA-KB tasks, the KB associated with SQ is chosen as FB2M, and that with WQ is chosen as the entire Freebase.

For both SQ and WQ, the subject mention μ in each question q is also annotated by matching the surface name of the subject entity with the words in q. Each answer triple (s, r, o) in SQ is presented directly in terms of the unique ID's specifying the involved entities and the unique string specifying the relation r. In WQ, however, for each answer path, the path label r is not given, and the entities s and o are only given in terms of their names. We therefore perform a simple search and screening to reconstruct each answer path in its unique (s, r, o) representation.

5.2 Implementations

Given the subject mention μ, the subject entity s in $\mathcal{K}$ can be found easily, as long as the question is well formed and the subject entity s exists in $\mathcal{K}$ unambiguously. Determining the subject entity s from subject mention μ usually only involves matching the words in u with the surface name string of the entities in $\mathcal{K}$. So the problem of entity alignment reduces to determining the subject mention μ in q. Conventionally, such a problem is treated as sequence labelling problem. In this work, our entity alignment model combines the recent BiGRU-CRF model (Dai et al., 2016) with joint character-level word encoding (Zhang et al., 2015). The entity alignment model is trained separately. In the model, the state/output dimension of the GRU network, the dimension of word embeddings and that of character encoding vectors are all set to 50.

The KSE model is also trained separately. In this model, FB2M is used to construct the schema graph for both SQ and WQ datasets, and the dimension of the embedding space is chosen as 100.

For ENC-DEC, the state dimension for the GRU network encoding u^{KB} and the input dimension are both chosen as 100. The state dimension for each of the two GRU networks for encoding u^{Q} is set to 200. Each input to these networks is a 200-dimensional word2vec vector (pre-trained on a large corpus WikiAnswers (Fader et al., 2013)) concatenated with a 10-dimensional position vector indicating the relative location of the word in the question. The state dimension for the GRU network in DEC is set to 300.

In CLS, the MLP has two ReLU activated latent layers, with dimensions 200 and 100 respectively.

Turbo training with mini-batch SGD is used to jointly train ENC-DEC and CLS. The batch size is set to 128. In the BASE phase, the ENC-DEC model is trained for 8000 batches on SQ, and 1500 batches on WQ. In the first round of TURBO phase, the number of batches in Steps A, B, and C are respectively 2000, 1000, 100 on SQ, and 1000, 500, 50 on WQ. In the remaining rounds, every step uses 100 batches on SQ, and 50 batches on WQ.

For the given question q, we denote the set of entities by $\mathcal{O}(s, r)$, such that there is a path in the KB graph that connect entity s to entity o via a path with label r. For a multiple-answer question q, not necessarily all entities in $\mathcal{O}(s, r)$ give rise to a correct answer. A usual approach to deal with this is to learn a separate binary classifier to screen the entities in $\mathcal{O}(s, r)$. In this work, we use a simple SVM classifier as the object answering model. The input feature vector for the SVM is the concatenation of three vectors: the average word2vec vectors of all words in q, the KSE embedding vector $\overline{r}_T$ of the final relation r_T in the path label r (assuming the path is T-hop), and the KSE embedding of $\bar{t}_o$ is the type $t(o)$ of the candidate object entity $o \in \mathcal{O}(s, r)$.

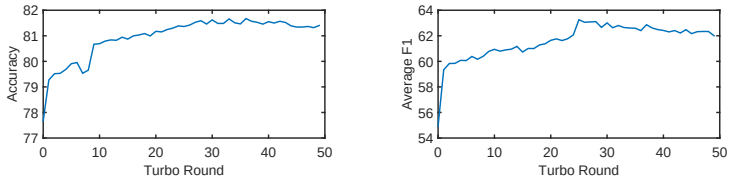
5.3 Question Answering Performance

Extensive experiments are performed to evaluate the proposed APVA-TURBO model in QA-KB tasks, relative to other existing models and various reductions of APVA-TURBO.

In experiments on SQ, the single answer triple (s, r, o) in $\widehat{A}(q)$ produced by a model is regarded as correct, if it matches the *unique* correct answer triple. The *accuracy* of a model is defined as the percentage of test questions for which the model produces the correct answer. On SQ, we use this accuracy to measure model performance.

On WQ, a given question q may have multiple answer paths (s, r, o) in the ground-truth answer set $\mathcal{A}(q)$. The model's *precision* for answering a question q is defined as the fraction of answers in $\widehat{\mathcal{A}}(q)$ that are in $\mathcal{A}(q)$, and the model's *recall* for answering q is the fraction of answers in $\mathcal{A}(q)$ that are in $\widehat{\mathcal{A}}(q)$. In such a setting, the overall performance is usually measured by the harmonic mean of precision and

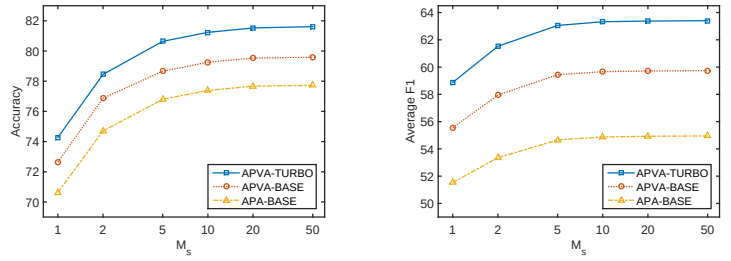
recall, namely, F1. We then use the average F1 value of a model over all test questions to measure the model's performance on WQ.



(a) SimpleQuestions (b) WebQuestions

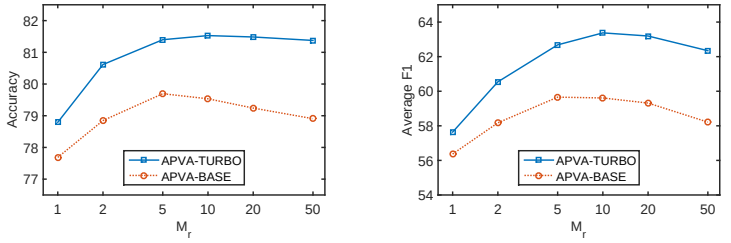
Figure 4: The performance of APVA-TURBO over TURBO iterations. $M_{\mathrm{s}} = 20$, $M_{\mathrm{r}} = 10$.

Analysis of APVA-TURBO The performance of APVA-TURBO over turbo iterations is shown in Figure 4. It can be seen that the performance increases with turbo iterations until 25 to 30 rounds. This demonstrates the effectiveness of turbo training. The performance starts to take a slight decreasing trend at around 30 to 40 rounds, due to overfitting, a phenomenon commonly arising when a model is overtrained. In the rest of the experiments, the number of Turbo rounds is taken as 30 for SQ and 25 for WQ.



(a) SimpleQuestions (b) WebQuestions

Figure 5: Performances of APVA-TURBO and its reductions vs M_{s}. $M_{\mathrm{r}} = 10$.



(a) SimpleQuestions (b) WebQuestions

Figure 6: Performances of APVA-TURBO and its reductions vs M_{r}. $M_{\mathrm{s}} = 20$.

Figure 5 plots the performances of APVA-TURBO and its various reductions as functions of the number M_{s} of candidate subject entities. APVA-BASE is the APVA-TURBO model without full turbo training. Specifically, after training in the Base Phase, it only enters the Turbo Phase once and stops right after the training of CLS. APA-BASE is a reduction of APVA-BASE by completely ignoring CLS. Including the verification classifier CLS appears to give a much significant boost of the performance (seen in APVA-BASE vs APA-BASE). Yet another leap in performance is achieved by including turbo training (seen in APVA-TURBO vs APVA-BASE).

It is also seen in Figure 5 that as M_{s} increases, the performances of the compared models all improve and eventually saturate after $M_{\mathrm{s}} = 20$. This performance improvement is expected since when more candidate subject entities are included, it increases the chance that the correct (s, r) pairs are selected.

In addition, although a high value of $M_{\rm s}$ also increases the chance of selecting a spurious (s, r) pair, we see no drop in performance. This is because the training of ENC-DEC has always been using *only the correct* subject entities. When a wrong entity is submitted to ENC-DEC, the path labels output from ENC-DEC tend to be a random draw, which are not competitive against the correct ones.

Figure 6 plots the performances of APVA-BASE and APVA-TURBO models against the number $M_{\rm r}$ of candidate path labels. Comparing the performance gap between the two models, it is apparent that turbo training brings in significantly benefit.

Figure 6 also shows that as $M_{\rm r}$ increases, the performances of the compared models all first increase and then gradually drop. This is because increasing $M_{\rm r}$ has two effects. The first is increasing the chance of selecting the correct path labels. The second is increasing the number of spurious path labels. It is sensible that a small increment of $M_{\rm r}$ makes the first effect dominate while a large increment makes the second dominate. The curves in the figure also show that the stronger a model is, at a higher $M_{\rm r}$ the performance trend reverts.

Table 1: Performance comparison on WQ

Model	F1(%)
(Yih et al., 2015)	52.5
(Xu et al., 2016)	53.3
(Yavuz et al., 2016)	52.6
(Jain, 2016)	**55.7**
(Reddy et al., 2017)	49.5
(Hao et al., 2017)	42.9
(Dong et al., 2017)	50.7
(Yue et al., 2017)	53.9
(Cheng et al., 2017)	50.1
APVA-BASE	**59.7**
APVA-TURBO	**63.4**

Table 2: Performance comparison on SQ

Model	Accuracy (%)
(Yin et al., 2016)	76.4
(Lukovnikov et al., 2017)	71.2
(Hsiao et al., 2017)	76.7
(Yu et al., 2017)	**78.7**
(Ture and Jojic, 2017)	**88.3**
APVA-BASE	**79.5**
APVA-TURBO	**81.5**

Performance Comparison with Existing Models The performance of APVA-TURBO is compared with the published performances of the existing models in Tables 1 and 2.

On WQ, the best known previous performance achieves an F1 value of 55.7%, due to (Jain, 2016). But this state of the art has been surpassed by APVA-BASE, with a F1 score 59.7%. This demonstrates the effectiveness of augmenting a model with a verification mechanism. With turbo training, APVA-TURBO further boosts its performance and presents a new F1 record of 63.4%. This performance, exceeding the current art by a stunning margin of nearly 8%, demonstrates the power of the APVA-TURBO approach developed in this paper.

On SQ, the best performance except that of (Ture and Jojic, 2017) achieves an accuracy of 78.7%. This performance has been surpassed by APVA-BASE, and further exceeded by APVA-TURBO at an accuracy of 81.5%. However that the current reported state of art is still that of (Ture and Jojic, 2017),

which achieves an accuracy of 88.3%. But that paper did not report sufficient details for us to understand in what respect APVA-TURBO under-performs. Neither does it contain all necessary details for us to implement the model and reproduce its reported performance.

6 Conclusion

This work demonstrates that incorporating a verification mechanism in a QA-KB model can be a powerful means of improving the model. Built on such a mechanism, the proposed APVA framework is also naturally facilitated with an iterative learning procedure, which we call turbo training. Turbo training is shown capable of significantly boosting the model performance. This makes APVA-TURBO a new state of art in QA-KB.

Acknowledgments

This work is supported partly by China 973 program (Nos.2015CB358700, 2014CB340305), by the National Natural Science Foundation of China (Nos.61772059, 61421003). This paper is also supported by the State Key Laboratory of Software Development Environment of China and Beijing Advanced Innovation Center for Big Data and Brain Computing.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *Computer Science*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning an executable neural semantic parser. *arXiv preprint arXiv:1711.05066*.

Zihang Dai, Lei Li, and Wei Xu. 2016. 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)*, volume 1, pages 800–810.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618. Association for Computational Linguistics.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Wei-Chuan Hsiao, Hen-Hsen Huang, and Hsin-Hsi Chen. 2017. Integrating subject, type, and property identification for simple question answering over knowledge base. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 976–985.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115.

Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. 2002. Rql: a declarative query language for rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 592–603. ACM.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Chimezie Ogbuji. 2005. Versa: Path-based rdf query language. *O'Reilly XML. com*.

Eric Prud, Andy Seaborne, et al. 2006. Sparql query language for rdf.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3104–3112. MIT Press.

Ferhan Ture and Oliver Jojic. 2017. No need to pay attention: Simple recurrent neural networks work! In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2866–2872.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2326–2336.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 149–159.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. 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)*, volume 1, pages 571–581.

Bin Yue, Min Gui, Jiahui Guo, Zhenglu Yang, Jin-Mao Wei, and Shaodi You. 2017. An effective framework for question answering over freebase via reconstructing natural sequences. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 865–866. International World Wide Web Conferences Steering Committee.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.