Hashing based Answer Selection

Dong Xu and Wu-Jun Li*
National Key Laboratory for Novel Software Technology
Collaborative Innovation Center of Novel Software Technology and Industrialization
Department of Computer Science and Technology, Nanjing University, China
dc.swind@gmail.com, liwujun@nju.edu.cn

Abstract

Answer selection is an important subtask of question answering (QA), in which deep models usually achieve better performance than non-deep models. Most deep models adopt question-answer interaction mechanisms, such as attention, to get vector representations for answers. When these interaction based deep models are deployed for online prediction, the representations of all answers need to be recalculated for each question. This procedure is time-consuming for deep models with complex encoders like BERT which usually have better accuracy than simple encoders. One possible solution is to store the matrix representation (encoder output) of each answer in memory to avoid recalculation. But this will bring large memory cost. In this paper, we propose a novel method, called hashing based answer selection (HAS), to tackle this problem. HAS adopts a hashing strategy to learn a binary matrix representation for each answer, which can dramatically reduce the memory cost for storing the matrix representations of answers. Hence, HAS can adopt complex encoders like BERT in the model, but the online prediction of HAS is still fast with a low memory cost. Experimental results on three popular answer selection datasets show that HAS can outperform existing models to achieve state-of-the-art performance.

Introduction

Question answering (QA) is an important but challenging task in natural language processing (NLP) area. Answer selection (answer ranking), which aims to select the corresponding answer from a pool of candidate answers for a given question, is one of the key components in many kinds of QA applications. For example, in community-based question answering (CQA) tasks, all answers need to be ranked according to the quality. In frequently asked questions (FAQ) tasks, the most related answers need to be returned back for answering the users' questions.

One main challenge of answer selection is that both questions and answers are not long enough in most cases. As a result, questions and answers usually lack background information and knowledge about the context (Deng et al. 2018). This phenomenon limits the performance of answer selection models. Deep neural networks (DNN) based models, also simply called deep models, can partly tackle this problem by using pre-trained word embeddings. Word embeddings pre-trained on language corpus contain some common knowledge and linguistic phenomena, which are helpful for selecting answers. Deep models have achieved promising performance for answer selection in recent years (Tan et al. 2016b; Santos et al. 2016; Tay, Tuan, and Hui 2018a; Tran and Niederée 2018; Deng et al. 2018).

Most deep models for answer selection are constructed with similar frameworks which contain an encoding layer (also called encoder) and a composition layer (also called composition module). Traditional models usually adopt convolutional neural networks (CNN) (Feng et al. 2015) or recurrent neural networks (RNN) (Tan et al. 2016b; Tran and Niederée 2018) as encoders. Recently, complex pre-trained models such as BERT (Devlin et al. 2018) and GPT-2 (Radford et al. 2019), are proposed for NLP tasks. BERT and GPT-2 adopt Transformer (Vaswani et al. 2017) as the key building block, which discards CNN and RNN entirely. BERT and GPT-2 are typically pre-trained on a large-scale language corpus, which can encode abundant common knowledge into model parameters. This common knowledge is helpful when BERT or GPT-2 is fine-tuned on other tasks.

The output of the encoder for each sentence of either question or answer is usually represented as a matrix and each column or row of the matrix corresponds to a vector representation for a word in the sentence. Composition modules are used to generate vector representations for sentences from the corresponding matrices. Composition modules mainly include pooling and question-answer interaction mechanisms. Question-answer interaction mechanisms include attention (Tan et al. 2016b), attentive pooling (Santos et al. 2016), multihop-attention (Tran and Niederée 2018) and so on. In general, question-answer interaction mechanisms have better performance than pooling. However, interaction mechanisms bring a problem that the vector representations of an answer are different with respect to different questions. When deep models with interaction mechanisms are deployed for online prediction, the representations of all answers need to be recalculated for each question. This pro-

*Wu-Jun Li is the corresponding author.

cedure is time-consuming for deep models with complex encoders like BERT which usually have better accuracy than simple encoders. One possible solution is to store the matrix representation (with float or double values) of each answer in memory to avoid recalculation. But this will bring large memory cost.

In this paper, we propose a novel method, called hashing based answer selection (HAS), to tackle this problem. The main contributions of HAS are briefly outlined as follows:

- HAS adopts a hashing strategy to learn a binary matrix representation for each answer, which can dramatically reduce the memory cost for storing the matrix representations of answers. To the best of our knowledge, this is the first time to use hashing for memory reduction in answer selection.
- By storing the (binary) matrix representations of answers in the memory, HAS can avoid recalculation for answer representations during online prediction. Subsequently, HAS can adopt complex encoders like BERT in the model, but the online prediction of HAS is still fast with a low memory cost.
- Experimental results on three popular answer selection datasets show that HAS can outperform existing models to achieve state-of-the-art performance.

Related Work

Answer Selection Most early models for answer selection are shallow (non-deep) models, which usually use bag-of-words (BOW) (Yih et al. 2013), term frequency (Robertson et al. 1994), manually designed rules (Téllez-Valero et al. 2011), syntactic trees (Wang and Manning 2010; Cui et al. 2005) as features. Different upper structures are designed for modeling the similarity of questions and answers based on these features. The main drawback of shallow models are the lacking of semantic information by using only surface features. Deep models can capture more semantic information by distributed representations, which lead to better results than shallow models. Early deep models use pooling (Feng et al. 2015) as the composition module to get vector representations for sentences from the encoder outputs which are represented as matrices. Pooling cannot model the interaction between questions and answers, which has been outperformed by new composition modules with question-answer interaction mechanisms. Attention (Bahdanau, Cho, and Bengio 2015) can generate a better representation of answers (Tan et al. 2016b) than pooling, by introducing the information flow between questions and answers into models. (Santos et al. 2016) proposes attentive pooling for bidirectional attention. (Tran and Niederée 2018) proposes a strategy of multihop attention which captures the complex relations between question-answer pairs. (Wan et al. 2016) focuses on the word by word similarity between questions and answers. (Wang, Liu, and Zhao 2016) and (Chen et al. 2018) propose inner attention which introduces the representation of question to the answer encoder through gates. (Tay, Tuan, and Hui 2018a) designs a cross temporal recurrent cell to model the interaction between questions and answers.

BERT and Transfer Learning To tackle the problem of insufficient background information and knowledge in answer selection, some methods introduce extra knowledge from other data. (Deng et al. 2018; Min, Seo, and Hajishirzi 2017; Wiese, Weissenborn, and Neves 2017) employ supervised transfer learning frameworks to pre-train a model from a source dataset. There are also some unsupervised transfer learning techniques (Yu et al. 2018; Chung, Lee, and Glass 2018). BERT (Devlin et al. 2018) is a recently proposed model for language understanding. By training on a large language corpus, abundant common knowledge and linguistic phenomena can be encoded into the parameters. As a result, BERT can be transferred to a wide range of NLP tasks and has shown promising results.

Hashing Hashing (Li, Wang, and Kang 2016) tries to learn binary codes for data representations. Based on the binary code, hashing can be used to speedup retrieval and reduce memory cost. In this paper, we take hashing to reduce memory cost, by learning binary matrix representations for answers. There have already appeared many hashing techniques for learning binary representation (Li, Wang, and Kang 2016; Cao et al. 2017; Hubara et al. 2016; Fan et al. 2019). To the best of our knowledge, there have not existed works to use hashing for memory reduction in answer selection.

Hashing based Answer Selection

In this section, we present the details of hashing based answer selection (HAS), which can be used to solve the problem faced by existing deep models with question-answer interaction mechanisms.

The framework of most existing deep models is shown in Figure 1(a). Compared with this framework, HAS has an additional *hashing layer*, which is shown in Figure 1(b). More specifically, HAS consists of an *embedding layer*, an *encoding layer*, a *hashing layer*, a *composition layer* and a *similarity layer*. With different choices of encoders (*encoding layer*) and composition modules (*composition layer*) in HAS, several different models can be constructed. Hence, HAS provides a flexible framework for modeling.

Embedding Layer and Encoding Layer

HAS is designed for modeling the similarity of question-answer pairs. Hence, the inputs to HAS are two sequences of words, corresponding to the question text and answer text respectively. Firstly, these sequences of words are represented by word embeddings through a word embedding layer. Suppose the dimension of word embedding is E, and the sequence length is L. The embeddings of question q and answer a are represented by matrices $\boldsymbol{Q}_q \in \mathbb{R}^{E \times L}$ and $\boldsymbol{A}_a \in \mathbb{R}^{E \times L}$ respectively. We use the same sequence length L for simplicity. Then, these two embedding matrices $\boldsymbol{Q}_q$ and $\boldsymbol{A}_a$ are fed into an *encoding layer* to get the contextual word representations. Different choices of *embedding layers* and encoders can be adopted in HAS. Here, we directly use the *embedding layer* and *encoding layer* in BERT to utilize the common knowledge and linguistic phenomena encoded in BERT. Hence, the formulation of *encoding layer* is as fol-

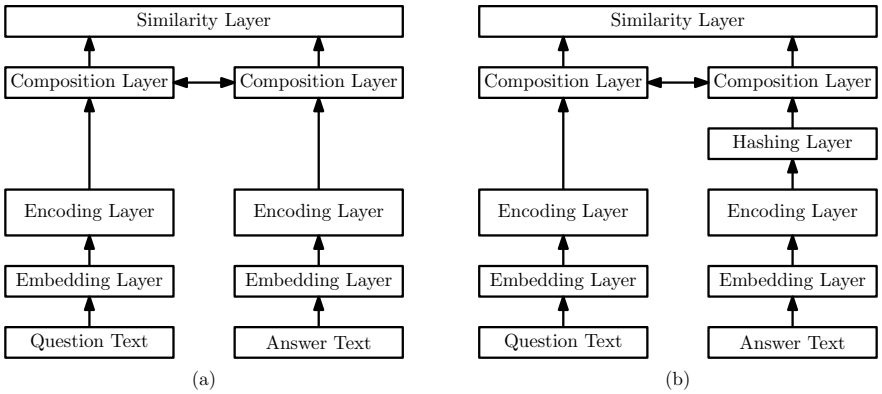


Figure 1: (a) Framework of traditional deep models for answer selection; (b) Framework of HAS.

lows:

$$\boldsymbol{U}_q = BERT(\boldsymbol{Q}_q),$$
$$\boldsymbol{V}_a = BERT(\boldsymbol{A}_a),$$

where $\boldsymbol{U}_q, \boldsymbol{V}_a \in \mathbb{R}^{D\times L}$ are the contextual semantic features of words extracted by BERT for question q and answer a respectively, and D is the output dimension of BERT.

Hashing Layer

The outputs of the *encoding layer* for question q and answer a are $\boldsymbol{U}_q$ and $\boldsymbol{V}_a$, which are two real-valued (float or double) matrices. When deep models with question-answer interaction mechanisms store the output of *encoding layer* ($\boldsymbol{V}_a$) in memory to avoid recalculation, they will meet the high memory cost problem. For example, if we take float values for $\boldsymbol{V}_a$, the memory cost for only one answer is over 600 KB when $L = 200$ and $D = 768$. Here, $D = 768$ is the output dimension of BERT. If the number of answers in candidate set is large, excessive memory cost will lead to impracticability, especially for mobile or embedded devices.

In this paper, we adopt hashing to reduce memory cost by learning binary matrix representations for answers. More specifically, we take the sign function $y = sgn(x)$ to binarize the output of the *encoding layer*. But the gradient of the sign function is zero for all nonzero inputs, which leads to a problem that the gradients cannot back-propagate correctly. $y = tanh(x)$ is a commonly used approximate function for $y = sgn(x)$, which can make the training process end-to-end with back-propagation (BP). Here, we use a more flexible variant $y = tanh(\beta x)$ with a hyper-parameter $\beta \geq 1$. The derivative of $y = tanh(\beta x)$ is

$$\frac{\partial y}{\partial x} = \beta(1 - y^2).$$

By using this function, the formulation of *hashing layer* is as follows:

$$\boldsymbol{B}_a = tanh(\beta \boldsymbol{V}_a), \tag{1}$$

where $\boldsymbol{B}_a \in \mathbb{R}^{D\times L}$ is the output of *hashing layer*.

To make sure that the elements in $\boldsymbol{B}_a$ can concentrate to binary values $\mathbb{B} = \{\pm 1\}$, we add an extra constraint for this layer as that in (Li, Wang, and Kang 2016):

$$\mathcal{J}^c(a) = ||\boldsymbol{B}_a - \boldsymbol{\mathcal{B}}_a||_F^2, \tag{2}$$

where $\boldsymbol{\mathcal{B}}_a \in \mathbb{B}^{D\times L}$ is the binary matrix representation for answer a, $||\cdot||_F$ is the Frobenius norm of a matrix. Here, $\boldsymbol{\mathcal{B}}_a$ is also a parameter to learn in HAS model.

When the learned model is deployed for online prediction, the learned binary matrices for answers will be stored in memory to avoid recalculation. With binary representation, each element in the matrices only costs one bit of memory. Hence, the memory cost can be dramatically reduced.

Composition Layer

The outputs of *encoding layer* and *hashing layer* are matrices of size $D \times L$. Composition layers are used to compose these matrix representations into vectors. Pooling, attention (Tan et al. 2016b), attentive pooling (Santos et al. 2016) and other interaction mechanisms (Tran and Niederée 2018; Wan et al. 2016) can be adopted in HAS. Interaction based modules usually have better performance than pooling based modules which have no question-answer interaction. Here, we take attention as an example to illustrate the advantage of HAS. More specifically, we adopt pooling for composing matrix representations of questions into question vectors, and adopt attention for composing matrix representations of answers into answer vectors. The formulation of the *composition layer* is as follows:

$$\boldsymbol{u}_q = max\_pooling(\boldsymbol{U}_q),$$
$$\boldsymbol{v}_a^{(q)} = attention(\boldsymbol{B}_a, \boldsymbol{u}_q) = \sum_{i=1}^{L} \alpha_i \cdot \boldsymbol{b}_i^{(a)},$$
$$\alpha_i \propto exp(\boldsymbol{m}^\top \cdot tanh(\boldsymbol{W}_1 \cdot \boldsymbol{b}_i^{(a)} + \boldsymbol{W}_2 \cdot \boldsymbol{u}_q)),$$

where $\boldsymbol{u}_q, \boldsymbol{v}_a^{(q)} \in \mathbb{R}^D$ are the composed vectors of questions and answers respectively, $\boldsymbol{b}_i^{(a)}$ is the i-th word representation in $\boldsymbol{B}_a = [\boldsymbol{b}_1^{(a)}, ..., \boldsymbol{b}_L^{(a)}]$, α_i is the attention weight for the i-th word which is calculated by a softmax function, $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{M\times D}$, $\boldsymbol{m} \in \mathbb{R}^M$ are attention parameters with M being the hidden size of attention.

The above formulation is for training. During test procedure, we just need to replace $\boldsymbol{B}_a$ by $\boldsymbol{\mathcal{B}}_a$.

Similarity Layer and Loss Function

The *similarity layer* measures the similarity between question-answer pairs based on their vector representations $\boldsymbol{u}_q$ and $\boldsymbol{v}_a^{(q)}$. Here, we choose cosine function as the similarity function, which is usually adopted in answer selection tasks:

$$s(q,a) = cos(\boldsymbol{u}_q, \boldsymbol{v}_a^{(q)}),$$

where $s(q,a) \in \mathbb{R}$ is the similarity between question q and answer a.

Based on the similarity between questions and answers, we can define the loss function. The most commonly used loss function for ranking is the triplet-based hinge loss (Tan et al. 2016b; Tran and Niederée 2018). To combine the hinge loss and the binary constraint in hashing together, we can get the following optimization problem:

$$\begin{aligned}\min_{\theta,\boldsymbol{\mathcal{B}}_\star} \mathcal{J} &= \sum_{(q,p,n)} [\mathcal{J}^m(q,p,n) + \delta \cdot \mathcal{J}^c(p) + \delta \cdot \mathcal{J}^c(n)] \\ &= \sum_{(q,p,n)} [max(0, 0.1 - s(q,p) + s(q,n)) + \\ &\qquad \delta \cdot ||\boldsymbol{B}_p - \boldsymbol{\mathcal{B}}_p||_F^2 + \delta \cdot ||\boldsymbol{B}_n - \boldsymbol{\mathcal{B}}_n||_F^2],\end{aligned}$$

where $\mathcal{J}^m(q,p,n) = max(0, 0.1 - s(q,p) + s(q,n))$ is the hinge loss for a triplet (q,p,n) from the training set, p is a positive answer corresponding to q, n is a randomly selected negative answer, δ is the coefficient of the binary constraint $\mathcal{J}^c(p)$ and $\mathcal{J}^c(n)$ for the positive answer p and the negative answer n respectively. $\boldsymbol{\mathcal{B}}_\star$ denotes a set of binary matrix representations for all answers. θ donates the parameters in HAS except $\boldsymbol{\mathcal{B}}_\star$.

These two sets of parameters θ and $\boldsymbol{\mathcal{B}}_\star$ can be optimized alternately (Li, Wang, and Kang 2016). More specifically, $\boldsymbol{\mathcal{B}}_a \in \boldsymbol{\mathcal{B}}_\star$ corresponding to answer a can be optimized as follows when θ is fixed:

$$\boldsymbol{\mathcal{B}}_a = sgn(\boldsymbol{B}_a).$$

And θ can be updated by utilizing back propagation (BP) when $\boldsymbol{\mathcal{B}}_\star$ is fixed.

Experiment

Datasets

We evaluate HAS on three popular answer selection datasets. The statistics about the datasets are presented in Table 1.

insuranceQA (Feng et al. 2015) is a FAQ dataset from insurance domain. We use the first version of this dataset, which has been widely used in existing works (Tan et al. 2016b; Wang, Liu, and Zhao 2016; Tan et al. 2016a; Deng et al. 2018; Tran and Niederée 2018). This dataset has already been partitioned into four subsets: Train, Dev, Test1 and Test2. The total size of candidate answers is 24981. To reduce the complexity, the dataset has provided a candidate set of 500 answers for each question, including positive and negative answers. There is more than one positive answer to some questions. As in existing works (Feng et al. 2015; Tran and Niederée 2018; Deng et al. 2018), we adopt Precision@1 (P@1) as the evaluation metric.

Table 1: Statistics of the datasets. "#questions" and "#C.A." denote the number of questions and candidate answers respectively.

	insuranceQA	yahooQA	wikiQA
#questions (Train)	12887	50112	873
#questions (Dev)	1000	6289	126
#questions (Test1)	1800	6283	243
#questions (Test2)	1800	—	—
#C.A. per question	500	5	9

yahooQA [1] is a large CQA corpus collected from Yahoo! Answers. We adopt the dataset splits as those in (Tay et al. 2017; Tay, Tuan, and Hui 2018a; Deng et al. 2018) for fair comparison. Questions and answers are filtered by their length, and only sentences with length among the range of 5 - 50 are preserved. The number of candidate answers for each question is five, in which only one answer is positive. The other four negative answers are sampled from the top 1000 hits using Lucene search for each question. As in existing works (Tay et al. 2017; Tay, Tuan, and Hui 2018a; Deng et al. 2018), P@1 and Mean Reciprocal Rank (MRR) are adopted as evaluation metrics.

wikiQA (Yang, Yih, and Meek 2015) is a benchmark for open-domain answer selection. The questions of wikiQA are factual questions which are collected from Bing search logs. Each question is linked to a Wikipedia page, and the sentences in the summary section are collected as the candidate answers. The size of candidate answer set for each question is different and there may be more than one positive answer to some questions. We filter out the questions which have no positive answers as previous works (Yang, Yih, and Meek 2015; Deng et al. 2018; Wang, Liu, and Zhao 2016). Mean Average Precision (MAP) and MRR are adopted as evaluation metrics as in existing works.

Hyperparameters and Baselines

We use base BERT as the encoder in our experiments. Large BERT may have better performance, but the *encoding layer* is not the focus of this paper. More specifically, the embedding size E and output dimension D of BERT are 768. The probability of dropout is 0.1. Weight decay coefficient is 0.01. Batch size is 64 for yahooQA, and 32 for insuranceQA and wikiQA. The attention hidden size M for insuranceQA is 768. M is 128 for yahooQA and wikiQA. Learning rate is $5e^{-6}$ for all models. The numbers of training epoches are 60 for insuranceQA, 18 for wikiQA and 9 for yahooQA. More epoches cannot bring apparent performance gain on the validation set. We evaluate all models on the validation set after each epoch and choose the parameters which achieve the best results on the validation set for final test. All reported results are the average of five runs.

There are also two other important parameters, β in $tanh(\beta x)$ and the coefficient δ of the binary constraint. β is tuned among $\{1, 2, 5, 10, 20\}$, and δ is tuned among $\{0, 1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$.

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&guccounter=1

Table 2: Results on insuranceQA. The results of models marked with ⋆ are reported from (Tran and Niederée 2018). Other results marked with ◇ are reported from their original paper. P@1 is adopted as evaluation metric by following previous works. 'our impl.' denotes our implementation.

Model	P@1 (Test1)	P@1 (Test2)
CNN ⋆	62.80	59.20
CNN with GESD ⋆	65.30	61.00
QA-LSTM (our impl.)	66.08	62.63
AP-LSTM ⋆	69.00	64.80
IARNN-GATE ⋆	70.10	62.80
Multihop-Sequential-LSTM ⋆	70.50	66.90
AP-CNN ◇	69.80	66.30
AP-BiLSTM ◇	71.70	66.40
MULT ◇	75.20	73.40
KAN (Tgt-Only) ◇	71.50	68.80
KAN ◇	75.20	72.50
HAS	**76.38**	**73.71**

The state-of-the-art baselines on three datasets are different. Hence, we adopt different baselines for comparison on different datasets according to previous works. Baselines using single model without extra knowledge include: CNN, CNN with GESD (Feng et al. 2015), QA-LSTM (Tan et al. 2016b), AP-LSTM (Tran and Niederée 2018), Multihop-Sequential-LSTM (Tran and Niederée 2018), IARNN-GATE (Wang, Liu, and Zhao 2016), NTN-LSTM, HD-LSTM (Tay et al. 2017), HyperQA (Tay, Tuan, and Hui 2018b), AP-CNN (Santos et al. 2016), AP-BiLSTM (Santos et al. 2016), CTRN (Tay, Tuan, and Hui 2018a), CA-RNN (Chen et al. 2018), RNN-POA (Chen et al. 2017), MULT (Wang and Jiang 2017), MV-FNN (Sha et al. 2018). Single models with external knowledge include: KAN (Deng et al. 2018). Ensemble models include: LRXNET (Narayan et al. 2018), $\text{SUM}_{BASE,PTK}$ (Tymoshenko and Moschitti 2018).

Because HAS adopts BERT as encoder, we also construct two BERT-based baselines for comparison. *BERT-pooling* is a model in which both questions and answers are composed into vectors by pooling. *BERT-attention* is a model which adopts attention as the composition module. Both BERT-pooling and BERT-attention use BERT as the encoder, and hashing is not adopted in them.

Experimental Results

Results on insuranceQA We compare HAS with baselines on insuranceQA dataset. The results are shown in Table 2. MULT (Wang and Jiang 2017) and KAN (Deng et al. 2018) are two strong baselines which represent the state-of-the-art results on this dataset. Here, KAN adopts external knowledge for performance improvement. KAN (Tgt-Only) denotes the KAN variant without external knowledge. We can find that HAS outperforms all the baselines, which proves the effectiveness of HAS.

Results on yahooQA We also evaluate HAS and baselines on yahooQA. Table 3 shows the results. KAN (Deng et al. 2018), which utilizes external knowledge, is the state-of-the-art model on this dataset. HAS outperforms all baselines except KAN. The performance gain of KAN mainly owes to the external knowledge, by pre-training on a source QA dataset SQuAD-T. Please note that HAS does not adopt external QA dataset for pre-training. HAS can outperform the target-only version of KAN, denoted as KAN (Tgt-Only), which is only trained on yahooQA without SQuAD-T. Once again, the result on yahooQA verifies the effectiveness of HAS.

Table 3: Results on yahooQA. The results of models marked with ⋆ are reported from (Tay, Tuan, and Hui 2018a). Other results marked with ◇ are reported from their original paper. P@1 and MRR are adopted as evaluation metrics by following previous works.

Model	P@1	MRR
Random Guess	20.00	45.86
NTN-LSTM ⋆	54.50	73.10
HD-LSTM ⋆	55.70	73.50
AP-CNN ⋆	56.00	72.60
AP-BiLSTM ⋆	56.80	73.10
CTRN ⋆	60.10	75.50
HyperQA ◇	68.30	80.10
KAN (Tgt-Only) ◇	67.20	80.30
KAN ◇	**74.40**	**84.00**
HAS	73.89	82.10

Table 4: Results on wikiQA. The results marked with ◇ are reported from their original paper. MAP and MRR are adopted as evaluation metrics by following previous works.

Model	MAP	MRR
AP-CNN ◇	68.86	69.57
AP-BiLSTM ◇	67.05	68.42
RNN-POA ◇	72.12	73.12
Multihop-Sequential-LSTM ◇	72.20	73.80
IARNN-GATE ◇	72.58	73.94
CA-RNN ◇	73.58	74.50
MULT ◇	74.33	75.45
MV-FNN ◇	74.62	75.76
$\text{SUM}_{BASE,PTK}$ ◇	75.59	77.00
LRXNET ◇	76.57	75.10
HAS	**81.01**	**82.22**

Results on wikiQA Table 4 shows the results on wikiQA dataset. $\text{SUM}_{BASE,PTK}$ (Tymoshenko and Moschitti 2018) and LRXNET (Narayan et al. 2018) are two ensemble models which represent the state-of-the-art results on this dataset. HAS outperforms all the baselines again, which further proves the effectiveness of our HAS.

Comparison with BERT-based Models We compare HAS with BERT-pooling and BERT-attention on three datasets. As shown in Table 5, BERT-attention and HAS outperform BERT-pooling on all three datasets, which verifies that question-answer interaction mechanisms have better performance than pooling. Furthermore, we can find that

Table 5: Comparison with BERT-based models.

	insuranceQA		yahooQA		wikiQA	
Model	P@1 (Test1)	P@1 (Test2)	P@1	MRR	MAP	MRR
BERT-pooling	74.52	71.97	73.49	81.93	77.22	78.27
BERT-attention	76.12	**74.12**	**74.78**	**82.68**	80.65	81.83
HAS	**76.38**	73.71	73.89	82.10	**81.01**	**82.22**

Table 6: Comparison of accuracy, time cost and memory cost on insuranceQA. Each question has 500 candidate answers. "Memory Cost $\diamond$ " is the memory cost for storing representations of answers.

Model	P@1 (Test1)	P@1 (Test2)	Time Cost per Question	Memory Cost $\diamond$
BERT-pooling	74.52	71.97	0.03s	0.07 GB
BERT-attention (recal.)	76.12	74.12	4.19s	0.02 GB
BERT-attention (store)	76.12	74.12	0.28s	14.29 GB
HAS	76.38	73.71	0.28s	0.45 GB
Multihop-Sequential-LSTM	70.50	66.90	0.13s	5.25 GB
AP-CNN	69.80	66.30	—	7.44 GB
AP-BiLSTM	71.70	66.40	—	5.25 GB

HAS can achieve comparable accuracy as BERT-attention. But BERT-attention has either speed (time cost) problem or memory cost problem, which will be shown in the following subsection.

We also find that HAS can improve the results of BERT-attention on insuranceQA and wikiQA. One reason might be that hashing can act as a regularization (constrained to be binary) for feature representation learning, and hence reduce the model complexity and increase the generalization ability when the model already has enough capacity. The wikiQA dataset is a relatively small dataset on which deep models are easy to overfit. HAS outperforms BERT-attention on wikiQA in terms of both MAP and MRR, which is consistent with our view about generalization.

Time Cost and Memory Cost To further prove the effectiveness of HAS, we compare HAS with baselines on insuranceQA in terms of time cost and memory cost when the model is deployed for prediction. The results are shown in Table 6. All experiments are run on a TitanXP GPU. BERT-pooling can directly store the vector representations of answers with a low memory cost, which doesn't have the time cost and memory cost problem. But the accuracy of BERT-pooling is much lower than BERT-attention and HAS. BERT-attention (recal.) denotes a BERT-attention variant which recalculates the matrix representations of answers for each question, and BERT-attention (store) denotes a BERT-attention variant which stores the matrix representations of answers in memory. BERT-attention (recal.) does not need to store the matrix representations of answers in memory, and BERT-attention (store) does not need recalculation. The time cost is 4.19 seconds per question for BERT-attention (recal.), which is about 15 times slower than HAS. Although BERT-attention (store) has low time cost as that of HAS, the memory cost of it is 14.29 GB, which is about 32 times larger than that of HAS.

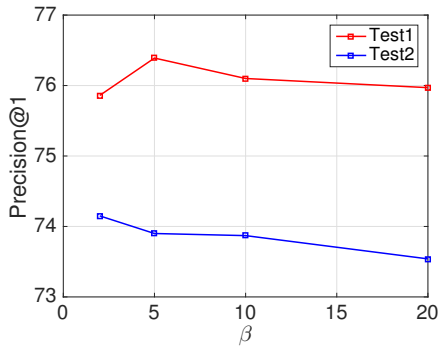
We also compare HAS with other baselines in existing works. KAN and MULT do question-answer interaction before *encoding layer* or during *encoding layer*, and the outputs of the *encoding layer* for an answer are different for different questions. Thus, these two models cannot store representations for reusing. We compare HAS with Multihop-Sequential-LSTM, AP-CNN, and AP-BiLSTM. The memory cost of these three models is 5.25 GB, 7.44 GB, and 5.25 GB, respectively, which are 11.75, 16.67, 11.75 times larger than that of HAS. Other baselines are not adopted for comparison, but almost all baselines with question-answer interaction mechanisms have either time cost problem or memory cost problem as that in BERT-attention.

We can find that our HAS is fast with a low memory cost, which also makes HAS have promising potential for embedded or mobile applications.

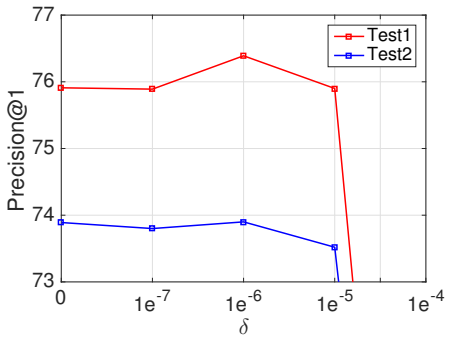
Sensitivity Analysis of δ and β In this section, we study the sensitivity of the two important hyper-parameters in HAS, which are the coefficient δ of $\mathcal{J}^c(a)$ and the value of β in $tanh(\beta x)$. We design a sensitivity study of these two hyper-parameters on insuranceQA and wikiQA. As shown in Figure 2(a) and Figure 3(a), the performance can be improved by increasing β to 5. We can find that HAS is not sensitive to β in the range of [5, 10]. When β is fixed to 5, the performance of different choices of δ is shown in Figure 2(b) and Figure 3(b). We can find that HAS is not sensitive to δ in the range of $[1e^{-7}, 1e^{-5}]$.

Conclusion

In this paper, we propose a novel answer selection method called hashing based answer selection (HAS). HAS adopts hashing to learn binary matrix representations for answers, which can dramatically reduce memory cost for storing the matrix outputs of encoders in answer selection. When deployed for prediction, HAS is fast with a low memory cost. This is particularly meaningful when the model needs to be deployed at embedded or mobile systems. Experimental results on three popular datasets show that HAS can outperform existing methods to achieve state-of-the-art performance.

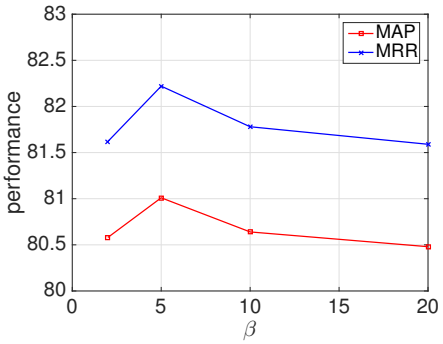


(a) Sensitivity of β when $\delta = 1e^{-6}$

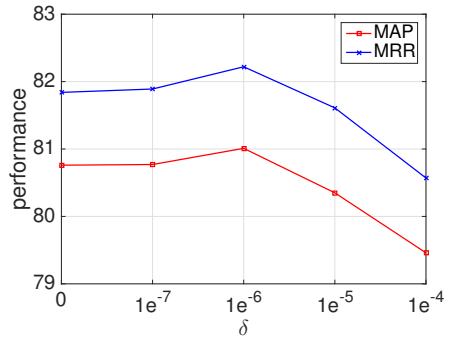


(b) Sensitivity of δ when $\beta = 5$

Figure 2: Sensitivity analysis on insuranceQA.



(a) Sensitivity of β when $\delta = 1e^{-6}$



(b) Sensitivity of δ when $\beta = 5$

Figure 3: Sensitivity analysis on wikiQA.

HAS is flexible to integrate other encoders and question-answer interaction mechanisms. Furthermore, the idea to adopt hashing for binary representation learning in HAS can also be used for other NLP tasks. All these possible extensions will be pursued in our future work.

Acknowledgments

This work is supported by the NSFC-NRF Joint Research Project (No. 61861146001).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. HashNet: Deep learning to hash by continuation. In *Proceedings of the IEEE International Conference on Computer Vision*, 5609–5618.

Chen, Q.; Hu, Q.; Huang, J. X.; He, L.; and An, W. 2017. Enhancing recurrent neural networks with positional attention for question answering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 993–996.

Chen, Q.; Hu, Q.; Huang, J. X.; and He, L. 2018. CA-RNN: Using context-aligned recurrent neural networks for modeling sentence similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 265–273.

Chung, Y.; Lee, H.; and Glass, J. R. 2018. Supervised and unsupervised transfer learning for question answering. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1585–1594.

Cui, H.; Sun, R.; Li, K.; Kan, M.; and Chua, T. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 400–407.

Deng, Y.; Shen, Y.; Yang, M.; Li, Y.; Du, N.; Fan, W.; and Lei, K. 2018. Knowledge as A bridge: Improving cross-domain answer selection with external knowledge. In *Proceedings of the International Conference on Computational Linguistics*, 3295–3305.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Fan, L.; Jiang, Q.-Y.; Yu, Y.-Q.; and Li, W.-J. 2019. Deep hashing for speaker identification and retrieval. In *Proceedings of the Annual Conference of the International Speech Communication Association*, 2908–2912.

Feng, M.; Xiang, B.; Glass, M. R.; Wang, L.; and Zhou, B. 2015. Applying deep learning to answer selection: A study and an open task. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 813–820.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 4107–4115.

Li, W.-J.; Wang, S.; and Kang, W.-C. 2016. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1711–1717.

Min, S.; Seo, M. J.; and Hajishirzi, H. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 510–517.

Narayan, S.; Cardenas, R.; Papasarantopoulos, N.; Cohen, S. B.; Lapata, M.; Yu, J.; and Chang, Y. 2018. Document modeling with external attention for sentence extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020–2030.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *Computing Research Repository*.

Robertson, S. E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.; and Gatford, M. 1994. Okapi at TREC-3. In *Proceedings of the Text REtrieval Conference*, 109–126.

Santos, C. d.; Tan, M.; Xiang, B.; and Zhou, B. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Sha, L.; Zhang, X.; Qian, F.; Chang, B.; and Sui, Z. 2018. A multi-view fusion neural network for answer selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5422–5429.

Tan, M.; dos Santos, C. N.; Xiang, B.; and Zhou, B. 2016a. Improved representation learning for question answer matching. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 464–473.

Tan, M.; Santos, C. d.; Xiang, B.; and Zhou, B. 2016b. LSTM-based deep learning models for non-factoid answer selection. In *Proceedings of the International Conference on Learning Representations*.

Tay, Y.; Phan, M. C.; Luu, A. T.; and Hui, S. C. 2017. Learning to rank question answer pairs with holographic dual LSTM architecture. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 695–704.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018a. Cross temporal recurrent networks for ranking question answer pairs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5512–5519.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018b. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 583–591.

Téllez-Valero, A.; Montes-y-Gómez, M.; Pineda, L. V.; and Padilla, A. P. 2011. Learning to select the correct answer in multi-stream question answering. *Information Processing and Management* 47(6):856–869.

Tran, N. K., and Niederée, C. 2018. Multihop attention networks for question answer matching. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 325–334.

Tymoshenko, K., and Moschitti, A. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2162–2173.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 6000–6010.

Wan, S.; Lan, Y.; Guo, J.; Xu, J.; Pang, L.; and Cheng, X. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2835–2841.

Wang, S., and Jiang, J. 2017. A compare-aggregate model for matching text sequences / semantic linking in convolutional neural networks for answer sentence selection. In *Proceedings of the International Conference on Learning Representations*.

Wang, M., and Manning, C. D. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the International Conference on Computational Linguistics*, 1164–1172.

Wang, B.; Liu, K.; and Zhao, J. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1288–1297.

Wiese, G.; Weissenborn, D.; and Neves, M. L. 2017. Neural domain adaptation for biomedical question answering. In *Proceedings of the Conference on Computational Natural Language Learning*, 281–289.

Yang, Y.; Yih, W.; and Meek, C. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013–2018.

Yih, W.; Chang, M.; Meek, C.; and Pastusiak, A. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1744–1753.

Yu, J.; Qiu, M.; Jiang, J.; Huang, J.; Song, S.; Chu, W.; and Chen, H. 2018. Modelling domain relationships for transfer learning on retrieval-based question answering systems in E-commerce. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 682–690.