Sequential Attention with Keyword Mask Model for Community-based Question Answering

Jianxin Yang[1,2], **Wenge Rong**[1,2], **Libin Shi**[3], **Zhang Xiong**[1,2]
[1]State Key Laboratory of Software Development Environment, Beihang University, China
[2]School of Computer Science and Engineering, Beihang University, China
[3]Microsoft, China
{yjx17,w.rong,xiongz}@buaa.edu.cn, libshi@microsoft.com

Abstract

In Community-based Question Answering system(CQA), Answer Selection(AS) is a critical task, which focuses on finding a suitable answer within a list of candidate answers. For neural network models, the key issue is how to model the representations of QA text pairs and calculate the interactions between them. We propose a Sequential Attention with Keyword Mask model(SAKM) for CQA to imitate human reading behavior. Question and answer text regard each other as context within keyword-mask attention when encoding the representations, and repeat multiple times(hops) in a sequential style. So the QA pairs capture features and information from both question text and answer text, interacting and improving vector representations iteratively through hops. The flexibility of the model allows to extract meaningful keywords from the sentences and enhance diverse mutual information. We perform on answer selection tasks and multi-level answer ranking tasks. Experiment results demonstrate the superiority of our proposed model on community-based QA datasets.

1 Introduction

Answering selection(AS) is one of the most fundamental challenges in community-based question answering(CQA) services. Given a question and a list of candidate answers, its aim is to choose the most matching one to the question. During this process of matching questions and answer candidates, how to encode the question and answer(QA) into meaningful and semantic representations impacts on the results directly.

Earlier conventional statistic methods are normally based on feature engineering and resource toolkits. Though these methods are easy in implementation, they require extra efforts and handcrafted features(Heilman and Smith, 2010; Tymoshenko and Moschitti, 2015). Recently, with the development of neural network, deep learning based models attract much attention in various tasks(Krizhevsky et al., 2012; Sutskever et al., 2014). In question answering field, the convolutional neural networks(CNNs)(Yu et al., 2014; Hu et al., 2014; Severyn and Moschitti, 2015) and recurrent neural networks(RNNs)(Wang and Nyberg, 2015; Feng et al., 2015) are widely employed to convert the question and answer text into vectors and define a feed-forward multi-layer perceptron to compute the interactions between them. These models construct sentences in an end-to-end fashion with less manual involvement. To capture fine-grained features, on the one hand, some works are concerned with matching QA pairs relationship in a more complex and diverse way, e.g., CNTN(Qiu and Huang, 2015) and MV-LSTM(Wan et al., 2016). On the other hand, latent representation models aim to jointly learn lexical and semantic information from QA sentences and influence the vector generation directly, e.g., attention mechanism(Bahdanau et al., 2015).

Attention mechanism learns attention weights of each words pairs between QA sentences. Afterwards it can calculate the weighted sum of hidden states over all time steps(dos Santos et al., 2016). This approach has shown promising results, while challenges still exist. For example, questions and answers in CQA services are generally long sentences, as such it is still difficult to compress all information into a fixed-length vector. To solve this problem, Sha et al. (2018) proposes co-attention view which brings improvement, and Zhang et al. (2018) further proposes a two-step attention to build dynamic question vectors based on various answer words. These kinds of methods usually require more parameters to learn representations. More importantly, when computing attention weights, every words

Proceedings of NAACL-HLT 2019, pages 2201–2211
Minneapolis, Minnesota, June 2 - June 7, 2019.

in QA pairs are involved. This word-to-word pattern takes meaningless noise into consideration, such as informal language usage or text irrelevant to the question. To alleviate this problem, Chen et al. (2018) proposes a context-aligned model to align phrase in QA relying on overlapped words and Stanford Core NLP tools[1]. Inspired by co-attention, we extend it to sequential style to learn better representations and try to extract useful keywords with less parameters and resource toolkits.

In this work, we propose a Sequential Attention with Keyword Mask(SAKM) model for answer selection task. We encode sentences similar to human reading behavior. When generating the question, our model refers to the answer and combines the mutual information. It is the same processing for producing answer representations. So when encoding a question, answer text is used as context and vice versa. We term this co-attention view as one "hop". Afterwards we repeat this process several times(hops) in a sequential style. As such QA pairs review each other recurrently to remind of mutual information and refine the sentence representations to be more precise across hops. Besides, the Keyword Mask modifies the attention mechanism such that the attention is computed over keywords instead of all words in the QA pair. So only keywords in the long context are extracted at each time step.

The contributions in this paper are three folders: 1) We extend attention mechanism to sequential structure, so the question and answer review each other recurrently to improve the sentence representations. 2) Different from standard soft attention, we propose sequential attention with keyword mask(SAKM) model. Besides, our model focuses on the significant words and filters other meaningless data. 3) We analyse the proposed SAKM model not only on classical answer selection tasks and but also multi-level answer ranking tasks. Experiment results show that our model tends to encode more rich semantic representations with less parameters.

2 Related Work

In Community-based Question Answering(CQA) services, since normally there exists a large number of question and answer pairs in repository, answer selection(AS) is a critical task, which focuses on finding a suitable answer within a list of candidate answers. As for traditional methods, feature engineering is a core work, but also a time-consuming and laborious task. BM25(Robertson et al., 1994) calculates relevance with item frequency, while language models(Ponte and Croft, 2017; Zhai and Lafferty, 2004) use the maximum likelihood of a word estimated from the question. Translation-based language models(Jeon et al., 2005; Xue et al., 2008) further improve. Considering the syntactic structure, some prior works(Heilman and Smith, 2010; Tymoshenko and Moschitti, 2015) convert the sentence into a tree structure by dependency parsing. Additionally, linguistics resources such as WordNet are utilized to enhance lexical features. Classification models like chain Conditional Random Fields have been used to match the questions and answers(Kiritchenko et al., 2014).

Recently, neural network based models have shown effectiveness in various fields, such as computer vision(Krizhevsky et al., 2012) and natural language processing(Kim, 2014). Different from aforementioned approaches, deep neural architectures(Hu et al., 2014; Wang and Nyberg, 2015) map each word into an embedding space, and compress the whole sentence into a low dimension vector. Then a similarity function is defined to calculate the interactions between QA pairs. Closer vectors in the embedding space represent much more relevant text.

To model fine-grained features, Qiu and Huang (2015) combines CNN with neural tensor network(NTN) to learn complex interactions between QA pairs. But NTN increases a lot of parameters and costs more runtime and memory. MV-LSTM proposed by Wan et al. (2016) uses bi-direction LSTM to generate a positional representation at each time step. Subsequently these representations from questions and answers are fed into a tensor layer. Shen et al. (2017) learns word representations in an embedding space by a translation matrix, and calculates relevance of each word pair in QA to compute a similarity matrix. Then CNN maps this matrix to a score scalar. Recently, (Tay et al., 2018b) applies the hyperbolic distance function to model the relationship between QA.

Other latent representation models construct interactions between QA when encoding sentences. More mutual information is extracted to learn a better latent representation. Miao et al. (2016) pro-

[1] https://stanfordnlp.github.io/CoreNLP/

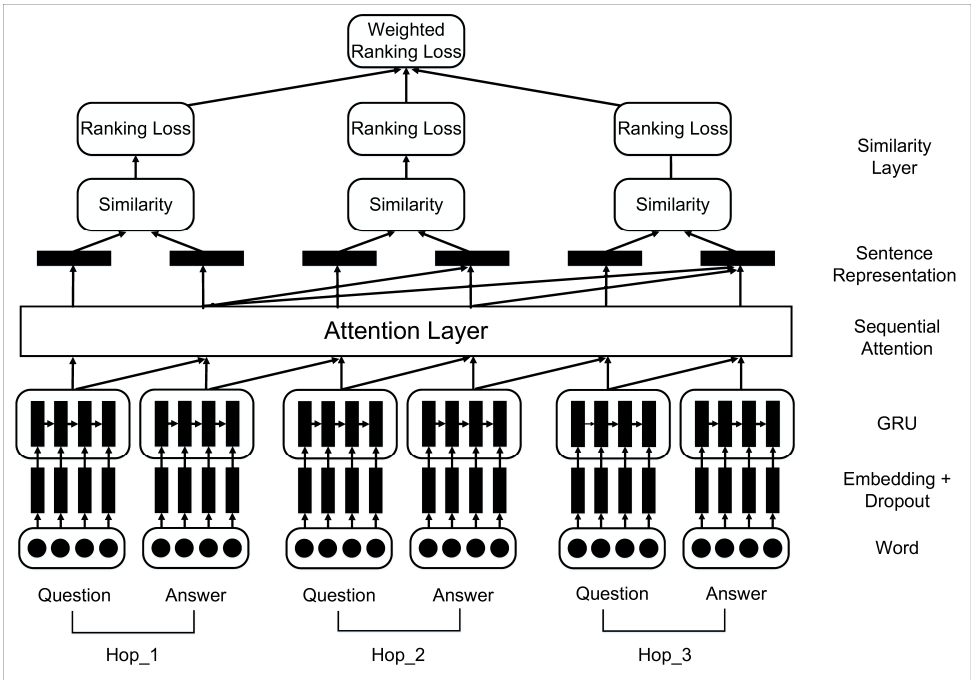


Figure 1: The architecture of 3 hops SAKM model. For simplification, we omit the lines in Sentence Representation Layer of Question part.

poses neural variational inference network. Wang et al. (2016) takes question context into consideration in RNN cell of answer network with gate mechanism. Yin et al. (2016) and dos Santos et al. (2016) propose attention-based CNN models to add attention weight matrix between a QA pair as a feature map. Additionally, Zhang et al. (2017) extends attention weight matrix to 3D tensor including more diverse information. Sha et al. (2018) proves co-attention view can significantly outperform the single attention. Further more, Zhang et al. (2018) constructs two-step attention to obtain question aware vectors based on various words in an answer sentence.

3 Sequential Attention with Keyword Mask Model

Given a question, which may contain one or more clauses, it can be denoted as $Q = (q_1, q_2, ..., q_n)$. Similarly, an answer can be denoted as $A = (a_1, a_2, ..., a_n)$. A^+ and A^- represent a positive answer and a negative answer, respectively. Fig. 1 describes the overall architecture of the proposed Sequential Attention with Keyword Mask(SAKM) model(in this figure, we use three hops as illustration). We extend our network in a sequential style. For each hop, a serial of stacked layers are constructed for the questions and the answers.

3.1 Embedding and Dropout

Firstly the questions and the answers need to be fed into the embedding layer and each word in sentences corresponds to an one-hot vector. Given a look-up table, each word is converted into an embedding space. The index of the low dimensional vector in the look-up table is the same as one-hot vector. We denote the embedding vectors of the QA pairs as $Q_{emb} = (x_1, x_2, ..., x_n) \in R^{d \times |Q|}$ and $A_{emb} = (y_1, y_2, ..., y_n) \in R^{d \times |A|}$, where d is the embedding size, $|Q|$ and $|A|$ denote the length of the question and answer respectively.

In order to mitigate the risk of overfitting, we employ dropout layer to randomly ignore different part of neurons in different hops during training. This process learns better representations of local regions and leads to better generalization during testing.

3.2 Sequential Attention

Gated Recurrent Unit(GRU) To encode a sentence into a single vector, we choose gated recurrent unit(Cho et al., 2014) and construct Q-GRU and A-GRU for the questions and answers, respectively. Given an input sentence $S = (s_1, s_2, ..., s_n) \in R^{d \times |S|}$, GRU handles each word recurrently and at time step t the operation is de-

fined as follow:

$$
\begin{aligned}
r_t &= \sigma(W_r s_t + U_r h_{t-1} + b_r) \\
z_t &= \sigma(W_z s_t + U_z h_{t-1} + b_z) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned} \tag{1}
$$

where

$$
\tilde{h}_t = \tanh\left(W_h s_t + U_h(r_t \odot h_{t-1}) + b_h\right) \tag{2}
$$

In the above equations, W_r, W_z, $W_h \in R^{m \times d}$ and U_r, U_z, $U_h \in R^{m \times m}$ are parameters in the neurons. m is the dimension size of the hidden states. b_r, b_z and $b_h \in R^m$ are bias. σ is sigmoid function and $\odot$ means element-wise product.

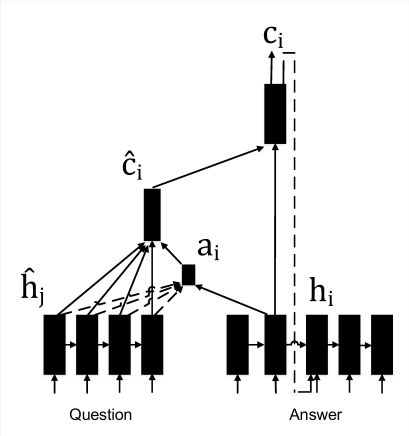


Figure 2: soft attention details.

Attention Mechanism During the GRU encoding process, an attention mechanism helps to combine context information with the current hidden states. For the standard soft attention mechanism(Bahdanau et al., 2015), as demonstrated in Fig. 2, the hidden state at time t computes attention weights with all of the context hidden states, and then obtains alignment scores after softmax operation. This mechanism takes all of the words in the context into consideration, while our model expects to extract some keywords to the current word and ignore other meaningless or noisy segments. The keyword mask relies on the attention weights and reserves top percent of words to account for alignment scores. It can be formulated as:

$$
\begin{aligned}
e_{ij} &= v^T \tanh(W_a[h_i; \hat{h}_j]) \\
e_{ij}^{mask} &= f_{mask\_top\_k}(e_{ij}, -inf) \\
a_{ij} &= \frac{\exp(e_{ij}^{mask})}{\sum_{k=1}^{|S|} \exp(e_{ik}^{mask})} \\
\hat{c}_i &= \sum_{j=1}^{|S|} a_{ij} \hat{h}_j \\
c_i &= \tanh(W_c[\hat{c}_i; h_i])
\end{aligned} \tag{3}
$$

where $[;]$ is the operator of concatenation, and $f_{mask\_top\_k}$ denotes the function that the top percent of values are reserved while others are masked as value $-inf$. So these masked positions in a_{ij} become 0 after softmax operation, which represents no influence to the current hidden state h_i. Fig. 3 shows the details of this attention mechanism. We will discuss the keywords percentage in more detail in Section 5.2.

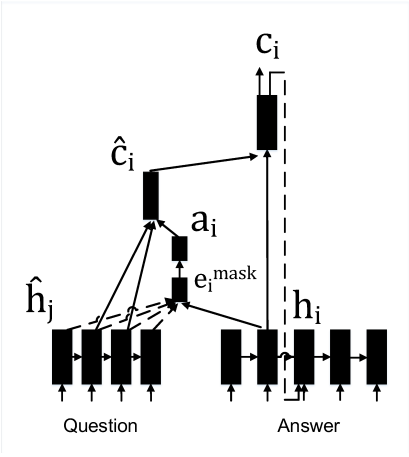


Figure 3: keyword-mask attention details.

After obtaining the representation of hidden neuron at time step t, we concat c_i with next word as input corresponding to the dotted line described in Fig. 3. When the whole sentence is processed, the final hidden output does not become representation directly because it loses much information about the beginning of the sentence. Instead, average operation over all hidden outputs is taken to produce the final representation.

Sequential Extension As shown in Fig. 1, A-GRU regards the question sentences as context to compute attention and representations. Likewise, Q-GRU reviews the answer contents to tune the representations. We extend this process in a sequential style to capture features and enhance information both from question text and answer text.

All of the parameters across hops are shared. For each hop, the vectors of QA pairs interact and improve. Compared to single direction attention or single hop attention, our model gets much more flexibility. It is capable of updating the representations towards the correct direction with the guide of a loss function and gradients across hops. We rename the Eq. 3 as $MaskAttention()$, as such the equations of the sequential extension is defined as:

$$\begin{aligned} Q^h &= MaskAttention(Q^h_{emb}, A^{h-1}) \\ A^h &= MaskAttention(A^h_{emb}, Q^h) \end{aligned} \tag{4}$$

where h is the current number of hop. The model iteratively updates the joint representations of the question and answer pair and obtains different outputs across hops.

Sentence Representation In the sentence representation layer, $Q^h_{representation}$ and $A^h_{representation}$ denote the final representation outputs of QA pairs at the hop h. Since Q^h and A^h only contain the information extracted at hop h, more meaningful content would be lost after more hops. But we expect to remember it from the beginning hops. To convey more information across hops, we do not simply take Q^h and A^h as the sentence representations. Instead, all of the previous outputs from $MaskAttention$ are involved. They can be calculated as:

$$\begin{aligned} Q^h_{representation} &= \frac{1}{h}\sum_{j=1}^{h} Q^h \\ A^h_{representation} &= \frac{1}{h}\sum_{j=1}^{h} A^h \end{aligned} \tag{5}$$

3.3 Similarity Calculation

Finally, we design a weighted loss strategy to compute the relevance and the loss value between QA pairs. For each hop, we have a pair of QA sentence representations and pass them through a similarity function described as:

$$s^h(Q,A) = \frac{{Q^h_{representation}}^T A^h_{representation}}{||Q^h_{representation}||_2 \cdot ||A^h_{representation}||_2} \tag{6}$$

where $s^h(Q,A)$ is the matching score between QA pairs at hop h. $||\cdot||_2$ means euclidean distance. As for the loss function during training, given a question, we use pair-wise margin-based ranking loss for a triple (Q, A^+, A^-). Thus the mathematical expression is:

$$\begin{aligned} L^h(Q,A^+,A^-) = \max(0, m - (s^h(Q,A^+) \\ - s^h(Q,A^-))) \end{aligned} \tag{7}$$

where m is the predefined margin.

Since we expect that vector representations generated from posterior hops are more precisely than the ones produced from prior hops, relatively small tolerance to the risk of matching incorrect QA pairs is accepted for posterior hops. Therefore, the loss values take increasing weights across hops. We denote r_h as loss weights. The objective loss function can be defined as:

$$L(Q,A^+,A^-) = \sum_{h=1}^{H} r_h L^h(Q,A^+,A^-) \tag{8}$$

4 Experimental Study

In this section, we test the proposed model on classical answer selection task and also multi-level answer ranking task to validate the model's effectiveness[2].

4.1 Answer Selection

Dataset & Implementation Details In this task, we use a community-based question answering dataset YahooCQA provided by Tay et al. (2017). It is an open-domain community forum, and the dataset contains 142,627 QA pairs. Sentences in YahooCQA are generally long and noisy. We follow the preprocessing in their work without extra process. Four negative answers are generated for a question using *Lucene*. Table 1 demonstrates the statistics of YahooCQA.

Dataset	YahooCQA	ZhihuCQA
# of Qns	50.1k / 6.2k / 6.2k	16k / 2k / 2k
# of Pairs	253k / 31.7k / 31.7k	80k / 10k / 10k

Table 1: Statistics of Datasets.

For our model, we tune the hidden size to 300, and the numbers of GRU layers for modeling questions and answers are both 1. Dropout is 0.5 and word embedding is pre-trained by skip-gram model. For the Sequential Extension layer, the number of hops is 3. For the Similarity Calculation, margin is 0.1 and weights for all hops are

[2]Our code is available at https://github.com/sheep-for/question_answer_matching

set as (0.2, 0.3, 0.5). Weights are set to constants because we promise to put more weight on later hops and keep reasonable tolerance for prior ones. Batch size is 20. All of the parameters are optimized by Back Propagation and Momentum.

Baselines We compare our model against several advanced deep neural network models. CNTN(Qiu and Huang, 2015), NTN-LSTM, HD-LSTM(Tay et al., 2017) and HyperQA(Tay et al., 2018b) are interaction focused methods, while AP-CNN, AP-BiLSTM(dos Santos et al., 2016), QRNN(Bradbury et al., 2017), CTRN(Tay et al., 2018a) and two-step attention(Zhang et al., 2018) are latent representation models. Additionally, we choose two traditional methods Random Guess and BM25(Robertson et al., 1994).

Evaluation Metrics For YahooCQA, we use Precision@1(P@1) and Mean Reciprocal Rank(MRR) to evaluate our model and the metrics are defined as:

$$
\begin{aligned}
P@1 &= \frac{1}{N}\sum_{i=1}^{N}\delta(r(A^{+})=1) \\
MRR &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r(q_i)}
\end{aligned}
\tag{9}
$$

where δ is indicator function, N is the number of all queries and $r(q_i)$ is the rank of the first correct answer to question q_i.

Model	P@1	MRR
Random Guess	20.0%	45.7%
BM25	22.5%	49.3%
CNN	41.3%	63.2%
LSTM	46.5%	66.9%
CNTN	46.5%	63.2%
NTN-LSTM	54.5%	73.1%
AP-CNN	56.0%	72.6%
AP-BiLSTM	56.8%	73.1%
QRNN	57.3%	73.6%
HD-LSTM	55.7%	73.5%
CTRN	60.1%	75.5%
Two-step attention	62.2%	77.4%
HyperQA	68.3%	80.1%
SA	66.0%	79.2%
SAKM	**69.3%**	**81.4%**

Table 2: Experiment results on YahooCQA. SA is the sequential attention baseline without keyword-mask operation. SAKM is our sequential attention with keyword mask model.

Experiment Results The results are shown in Table 2. Firstly, it is observed that deep neural network models outperform traditional models. Most latent representation models obtain better results than interaction focused models, indicating that earlier interactions when encoding sentences produces semantic vectors. Most importantly, the proposed SAKM model achieves best results on both P@1 and MRR. Our basic SA model outperforms two-step attention model by 3.8% in terms of P@1 and 1.8% in terms of MRR, which shows that our sequential extension structure is effective. Furthermore, our SAKM model outperforms HyperQA model by 1.0% in terms of P@1 and 1.3% in terms of MRR. Since HyperQA is an interaction focused model which adopts the hyperbolic distance function to model the relevance between QA. we can combine it with our SAKM to obtain better performance in further study. The experiment results agree with our intuition that extracting meaningful keywords in attention mechanism helps to generate more precise representations.

4.2 Multi-Level Answer Ranking

Relevant relationship in answer selection datasets is binary, only including relevance and irrelevance. However, in the real CQA applications, it is difficult to verify whether the answers are completely correct or not. This scenario has caused a challenge called multi-level answer ranking(Liu et al., 2018). These answers for one question are annotated as several levels corresponding to the thumb-up numbers.

Dataset & Implementation Details To test the proposed model in multi-level answer ranking task, we choose the dataset ZhihuCQA provided by Liu et al. (2018). Zhihu[3] is a popular and professional Chinese QA community platform with more than millions of users and QA pairs. Table 1 describes the statistics of ZhihuCQA. For each question, top five answers are selected and ranked by the thumb-up numbers. We replace margin based ranking loss with RankNet(Burges et al., 2005).

In this task, we use the jieba[4] toolkits for word segmentation and tune the hidden size to 200, and the numbers of GRU layers for modeling questions and answers are both 2. Other settings are the same as YahooCQA.

[3] https://www.zhihu.com/
[4] https://github.com/fxsjy/jieba

Baselines We compare our model against available advanced methods in (Liu et al., 2018). ARC-II learns hierarchical pattern based on ARC-I(Hu et al., 2014). Skip-Thoughts model(Kiros et al., 2015) trains an encoder-decoder model to construct sentence vectors. Attentive LSTM, ABCNN and Compare-Aggregate(Wang and Jiang, 2017) are attention-based models and Rewrite+Rank is based on generative adversarial network.

Evaluation Metrics In this task, since the labels for an answer are not binary, we choose normalized discounted cumulative gain(NDCG) and expected reciprocal rank(ERR) for evaluation. $(o_1, o_2, ..., o_M)$ denotes the predicted orders of answers to a question. NDCG is defined as:

$$\begin{aligned} NDCG &= \frac{DCG}{iDCG} \\ DCG &= \sum_{i=1}^{M} \frac{2^{o_i} - 1}{\log(1+i)} \end{aligned} \tag{10}$$

where $iDCG$ is the ideal DCG calculated from the correct orders $(l_1, l_2, ..., l_M)$. ERR is defined as:

$$\begin{aligned} ERR &= \sum_{r=1}^{M} \frac{R_r}{r} \prod_{i=1}^{r-1}(1 - R_i) \\ R_i &= \frac{2^{o_i} - 1}{2^{o_m}} \end{aligned} \tag{11}$$

where o_m is the maximum of degree values.

Model	NDCG	ERR
Random Guess	41.7%	30.3%
ARC-I	64.4%	56.4%
ARC-II	68.5%	58.8%
Skip-Thoughts	69.1%	60.1%
Attentive LSTM	71.4%	61.1%
ABCNN	72.5%	62.0%
Compare-Aggregate	74.8%	63.2%
Rewrite+Rank	77.2%	65.0%
SA	80.7%	68.2%
SAKM	**81.0%**	**68.4%**

Table 3: Experiment results on ZhihuCQA.

Experiment Results Table 3 reports the results on ZhihuCQA. Similar to YahooCQA, attention-based models perform better than other baselines. Our SA model outperforms Rewrite+Rank by 3.6% in terms of NDCG and 3.2% in terms of ERR. SAKM model achieves slightly improvement compared to SA version. Results show that on multi-level answer ranking task, our review mechanism allows question and answer interaction while encoding in a more fine-grain aspect and leads to better performance.

5 Discussion and Analysis

In this section, we divide our discussion into three parts, including the trade-off between information transmission and avoidance of overfitting, the relationship between sentence length and keyword percentage, the advantages of our SAKM model.

5.1 Trade-off between Information Transmission and Avoidance of Overfitting

Our model processes QA text in a sequential style. For the first hop, the original contents are fed as inputs. Afterwards, the representations are updated based on previous outputs across hops, thus it is significant to convey rich mutual information. Meanwhile, since the sentences are long and redundant, it is necessary to avoid overfitting.

Information Transmission across Hops In order to utilize context better, We propose sequential style to refine the sentence vectors through multiple hops. When calculating the sentence representations, we take the average over outputs of all time steps instead of selecting the final hidden state, and get the final sentence representations based on all hops. Additionally, in embedding layer we pretrain the word embeddings on the corpus using word2vec(Mikolov et al., 2013). To guide the vectors update in a correct direction based on gradients, the loss function is calculated over all hops, and puts more weight on later ones.

Avoidance of Overfitting There are some tricks to reduce the risk of overfitting. Our model extracts some keywords according to the attention weights, and applies dropout to ignore different neurons in different hops. Besides, the GRU layer is shallow and the number of hops is suitable.

5.2 Relationship between Sentence Length and Keyword Percentage

In the attention mechanism, we use $f_{mask\_top\_k}$ to reserve top percent of attention weights. In this part, we propose two strategies to explore the relationship between the sentence length and keyword percentage.

Fixed-Percentage: We set the number of keywords based on the statistics of the dataset. Firstly,

we count the lengths of all questions and answers respectively, and then sort them in an ascending order. We choose the value of the third quartile as the number of keywords in a question, while the value of the first quartile as the number of keywords in an answer. This strategy allows us not to calculate percentage according to various lengths in the dataset. Our experiments empirically show that it works well.

Variable-Percentage: The second strategy is to compute the number of keywords for all sentences.

Question: Since the answers are produced based on question sequences, the question generally contains more meaningful information such as inquiry type, inquiry main verb, topic and so on. We empirically calculate the number of keywords k in a question of length x as follow:

$$k = \min(10 \ln x, x) \tag{12}$$

Answer: As for answers submitted by users in community forum, there exists redundant contents, typos errors, emoticon and other informal language usage. Inspired by TF-IDF algorithm, in our experiments, we propose a heuristics rule to calculate the number of keywords k. The length of an answer is denoted as x. For the first part, we compute the Total-Length(TL) term as follow:

$$TL(x) = x\lfloor \lg x \rfloor \tag{13}$$

where $\lfloor \cdot \rfloor$ is rounding down operation. TL value increases monotonously with length of sentence. For the second part, we compute the Inverse-Noise-Frequency(INF) term as follow:

$$INF(x) = \frac{1}{\lg 2x} \tag{14}$$

This term represents the percent of the meaningful words, which is an inverse proportion to noisy words. Finally, the number of keywords k can be obtained by multiplying these two terms.

$$k = \min(TL(x) \cdot INF(x), x) \tag{15}$$

5.3 Advantages of SAKM model

Simplicity: Our SAKM model is simple but outperforms on large CQA datasets. The network is shallow and all of the parameters across hops are shared. Our model is not complicated and has less parameters than other mentioned neural network models. Table 4 demonstrates the complexity analysis of some models.

Our model is an end-to-end neural network, and trained via back-propagation automatically. The SAKM model could be an universal way to learn sentence vectors effectively and integrated in other larger neural network models. It could be a useful tool in building neural architecture based representations for text sequences.

Model	Complexity	Parameters
AP-BiLSTM	$4(md + d^2) + 4d^2$	1.08M
NTN	$2dk + 2k + d^2k$	2.1M
CTRN	$3kdm + 2dh + h$	1.05M
Two-step	$6md + 10d^2 + 12d^2k$	1.31M
SAKM (Our)	$3(md + d^2) + 4d^2$	0.9M

Table 4: Complexity analysis on YahooCQA. m is embedding size, k is the filter width, d is the output dimension and h is the size of full-connected layer.

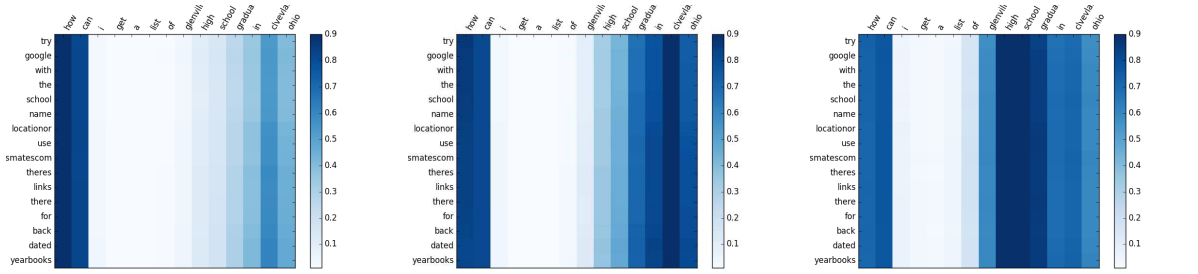
Convergency: As the sentence representations are tuned and improved across hops in one epoch, it costs less epochs for our model to converge. In our experiments, performance improve quickly in first ten epochs.

Effectiveness: The QA pairs capture features and information both from question text and answer text, iteratively updating and improving question and answer representations through hops.

At the test time, choosing the outputs of the last hop from sentence representation layer as sentence vectors can obtain better results on P@1 and MRR, demonstrating the effectiveness of improvement across hops. SAKM model outperforms SA model by more than 3% on P@1 and 2% on MRR. It proves that extracting keywords is significant and necessary. Besides, even if we choose the first hop of SA model, the gains are significant compared to two-way attention model. It means that our refinement procedure leads to better representations for all hops. Table 5 shows the details.

Representation	P@1	MRR
SA_hop_1	64.5%	78.2%
SA_hop_last	66.0%	79.2%
SA_concat_hops	65.7%	79.0%
SAKM_hop_1	68.7%	81.0%
SAKM_hop_last	**69.3%**	**81.4%**
SAKM_concat_hops	69.2%	81.3%

Table 5: Experiment results on YahooCQA using various representations. SAKM_hop_1 denotes that SAKM model tests with the sentence representations of the first hop.



(a) Visualization of SA at first hop (b) Visualization of SA at last hop (c) Visualization of SAKM at last hop

Figure 4: Attention Visualization.

Given a pair of QA as example. *Q: How can i get a list of glenville high school graduates in clvevland ohio. A: Try google with the school name locationor use classmatescom theres links there for back dated yearbooks.* Fig. 4 displays the heatmap of the attention weights. We can observe that compared to the first hop of SA model, the last hop puts more weights on phrase *glenville high school graduates in clvevland ohio*. Further more, the last hop of SAKM model focuses on keywords *how can, glenville high school graduates.* It shows that the inquiry type words and topic words achieve more attention. From this heatmap, it indicates that our SAKM model is reasonable and works well.

6 Conclusion

In this work, we propose a sequential attention with keyword mask model for CQA. Our model handles answer selection task similar to human reading behavior. The questions and answers review each other recurrently to improve the representations. This proposed attention mechanism focuses on some keywords and filters other meaningless data. We evaluate our model on two tasks, answering selection and multi-level answer ranking. The experiment results demonstrate that our model outperforms on CQA datasets and enhance mutual information between QA pairs effectively.

Acknowledgments

This work was supported in part by the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2019ZX-16).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of 2015 International Conference on Learning Representations*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. *Proceedings of 2017 International Conference on Learning Representations*.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of 22nd International Conference on Machine Learning*, pages 89–96.

Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, and Liang He. 2018. CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 265–273.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Proceedings of 2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 813–820.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of 2010 Conference of the North American Chapter of the Association of Computational Linguistics*, pages 1011–1019.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of 2014 Annual Conference on Neural Information Processing Systems*, pages 2042–2050.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management*, pages 84–90.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif M Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 437–442.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of 2015 Annual Conference on Neural Information Processing Systems*, pages 3294–3302.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of 2012 Annual Conference on Neural Information Processing Systems*, pages 1097–1105.

Yang Liu, Wenge Rong, and Zhang Xiong. 2018. Improved text matching by enhancing mutual information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of 2016 International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Wentau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of 2013 Conference of the North American Chapter of the Association of Computational Linguistics*, pages 746–751.

Jay M. Ponte and W. Bruce Croft. 2017. A language modeling approach to information retrieval. *SIGIR Forum*, 51(2):202–208.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1305–1311.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline Hancockbeaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference*, pages 109–126.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382.

Lei Sha, Jinge Yao, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. A multi-view fusion neural network for answer selection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5422–5429.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. 2017. Word embedding based correlation model for question/answer matching. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3511–3517.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of 2014 Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. Cross temporal recurrent networks for ranking question answer pairs. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5512–5519.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 583–591.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1451–1460.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2835–2841.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 707–712.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of 2017 International Conference on Learning Representations*.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Chengxiang Zhai and John D Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Pengqing Zhang, Yuexian Hou, Zhan Su, and Yi Su. 2018. Two-step multi-factor attention neural network for answer selection. In *Proceedings of 15th Pacific Rim International Conference on Artificial Intelligence*, pages 658–670.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive interactive neural networks for answer selection in community question answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3525–3531.