Learning to Rank Question-Answer Pairs using Hierarchical Recurrent Encoder with Latent Topic Clustering

Seunghyun Yoon, Joongbo Shin and Kyomin Jung
Dept. of Electrical and Computer Engineering
Seoul National University, Seoul, Korea
{mysmilesh,jbshin,kjung}@snu.ac.kr

Abstract

In this paper, we propose a novel end-to-end neural architecture for ranking candidate answers, that adapts a hierarchical recurrent neural network and a latent topic clustering module. With our proposed model, a text is encoded to a vector representation from an word-level to a *chunk-level* to effectively capture the entire meaning. In particular, by adapting the hierarchical structure, our model shows very small performance degradations in longer text comprehension while other state-of-the-art recurrent neural network models suffer from it. Additionally, the latent topic clustering module extracts semantic information from target samples. This clustering module is useful for any text related tasks by allowing each data sample to find its nearest topic cluster, thus helping the neural network model analyze the entire data. We evaluate our models on the Ubuntu Dialogue Corpus and consumer electronic domain question answering dataset, which is related to Samsung products. The proposed model shows state-of-the-art results for ranking question-answer pairs.

1 Introduction

Recently neural network architectures have shown great success in many machine learning fields such as image classification, speech recognition, machine translation, chat-bot, question answering, and other task-oriented areas. Among these, the automatic question answering (QA) task has long been considered a primary objective of artificial intelligence.

In the commercial sphere, the QA task is usually tackled by using pre-organized knowledge bases and/or by using information retrieval (IR) based methods, which are applied in popular intelligent voice agents such as *Siri*, *Alexa*, and *Google Assistant* (from Apple, Amazon, and Google, respectively). Another type of advanced QA systems is IBM's Watson who builds knowledge bases from unstructured data. These raw data are also indexed in search clusters to support user queries (Fan et al., 2012; Chu-Carroll et al., 2012).

In academic literature, researchers have intensely studied sentence pair ranking task which is core technique in QA system. The ranking task selects the best answer among candidates retrieved from knowledge bases or IR based modules. Many neural network architectures with end-to-end learning methods are proposed to address this task (Yin et al., 2016; Wang and Jiang, 2016; Wang et al., 2017). These works focus on matching sentence-level text pair (Wang et al., 2007; Yang et al., 2015; Bowman et al., 2015). Therefore, they have limitations in understanding longer text such as multi-turn dialogue and explanatory document, resulting in performance degradation on ranking as the length of the text become longer.

With the advent of the huge multi-turn dialogue corpus (Lowe et al., 2015), researchers have proposed neural network models to rank longer text pair (Kadlec et al., 2015; Baudiš et al., 2016). These techniques are essential for capturing context information in multi-turn conversation or understanding multiple sentences in explanatory text.

In this paper, we focus on investigating a novel neural network architecture with additional data clustering module to improve the performance in ranking answer candidates which are longer than a single sentence. This work can be used not only for the QA ranking task, but also to evaluate the relevance of next utterance with given dialogue generated from the dialogue model. The key contributions of our work are as follows:

First, we introduce a Hierarchical Recurrent Dual Encoder (HRDE) model to effectively calculate the affinity among question-answer pairs to determine the ranking. By encoding texts from an word-level to a chunk-level with hierarchi-

Proceedings of NAACL-HLT 2018, pages 1575–1584
New Orleans, Louisiana, June 1 - 6, 2018.

cal architecture, the HRDE prevents performance degradations in understanding longer texts while other state-of-the-art neural network models suffer.

Second, we propose a Latent Topic Clustering (LTC) module to extract latent information from the target dataset, and apply these additional information in end-to-end training. This module allows each data sample to find its nearest topic cluster, thus helping the neural network model analyze the entire data. The LTC module can be combined to any neural network as a source of additional information. This is a novel approach using latent topic cluster information for the QA task, especially by applying the combined model of HRDE and LTC to the QA pair ranking task.

Extensive experiments are conducted to investigate efficacy and properties of the proposed model. Our proposed model outperforms previous state-of-the-art methods in the Ubuntu Dialogue Corpus, which is one of the largest text pair scoring datasets. We also evaluate the model on real world QA data crawled from crowd-QA web pages and from Samsung's official web pages. Our model also shows the best results for the QA data when compared to previous neural network based models.

2 Related Work

Researchers have released question and answer datasets for research purposes and have proposed various models to solve these datasets. (Wang et al., 2007; Yang et al., 2015; Tan et al., 2015) introduced small dataset to rank sentences that have higher probabilities of answering questions such as WikiQA and insuranceQA. To alleviate the difficulty in aggregating datasets, that are large and have no license restrictions, some researchers introduced new datasets for sentence similarity rankings (Baudiš et al., 2016; Lowe et al., 2015). As of now, the Ubuntu Dialogue dataset is one of the largest corpus openly available for text ranking.

To tackle the Ubuntu dataset, (Lowe et al., 2015) adopted the "term frequency-inverse document frequency" approach to capture important words among context and next utterances (Ramos et al., 2003). (Bordes et al., 2014; Yu et al., 2014) proposed deep neural network architecture for embedding sentences and measuring similarities to select answer sentence for a given question. (Kadlec et al., 2015) used convolution neural network (CNN) architecture to embed the sentence while a final output vector was compared to the target text to calculate the matching score. They also tried using long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), bidirectional LSTM and ensemble method with all of those neural network architectures and achieved the best results on the Ubuntu Dialogues Corpus dataset. Another type of neural architecture is the RNN-CNN model, which encodes each token with a recurrent neural network (RNN) and then feeds them to the CNN (Baudiš et al., 2016). Researchers also introduced an attention based model to improve the performance (Tan et al., 2015; Wang and Jiang, 2016; Wang et al., 2017).

Recently, the hierarchical recurrent encoder-decoder model was proposed to embed contextual information in user query prediction and dialogue generation tasks (Sordoni et al., 2015; Serban et al., 2016). This shows improvement in the dialogue generation model where the context for the utterance is important. As another type of neural network architecture, memory network was proposed by (Sukhbaatar et al., 2015). Several researchers adopted this architecture for the reading comprehension (RC) style QA tasks, because it can extract contextual information from each sentence and use it in finding the answer (Xiong et al., 2016; Kumar et al., 2016). However, none of this research is applied to the QA pair ranking task directly.

3 Model

In this section, we depict a previously released neural text ranking model, and then introduce our proposed neural network model.

3.1 Recurrent Dual Encoder (RDE)

A subset of sequential data is fed into the recurrent neural network (RNN) which leads to the formation of the network's internal hidden state h_t to model the time series patterns. This internal hidden state is updated at each time step with the input data w_t and the hidden state of the previous time step h_{t-1} as follows:

$$h_t = f_\theta(h_{t-1}, w_t), \tag{1}$$

where f_θ is the RNN function with weight parameter θ, h_t is hidden state at t-th word input, w_t is t-th word in a target question $\boldsymbol{w}^Q = \{w^Q_{1:t_q}\}$ or an answer text $\boldsymbol{w}^A = \{w^A_{1:t_a}\}$.

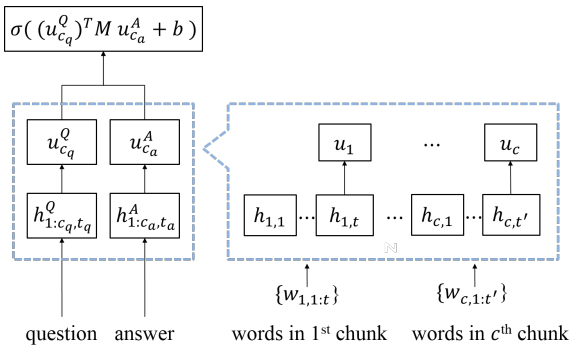

Figure 1: Diagram of the HRDE model. The word-lever RNN encodes words sequences of each chunk. The the final hidden status of the word-level RNN is fed into chunk-level RNN.

The previous RDE model uses two RNNs for encoding question text and answer text to calculate affinity among texts (Lowe et al., 2015). After encoding each part of the data, the affinity among the text pairs is calculated by using the final hidden state value of each question and answer RNNs. The matching probability between question text $\boldsymbol{w}^Q$ and answer text $\boldsymbol{w}^A$ with the training objective are as follows:

$$
\begin{aligned}
p(\text{label}) &= \sigma((h_{t_q}^Q)^T M\ h_{t_a}^A + b), \\
\mathcal{L} &= -\log \prod_{n=1}^{N} p(\text{label}_n | h_{n,t_q}^Q, h_{n,t_a}^A),
\end{aligned}
\tag{2}
$$

where $h_{t_q}^Q$ and $h_{t_a}^A$ are last hidden state of each question and answer RNN with the dimensionality $h_t \in \mathbb{R}^d$. The $M \in \mathbb{R}^{d \times d}$ and bias b are learned model parameters. The N is total number of samples used in training and σ is the sigmoid function.

3.2 Hierarchical Recurrent Dual Encoder (HRDE)

From now we explain our proposed model. The previous RDE model tries to encode the text in question or in answer with RNN architecture. It would be less effective as the length of the word sequences in the text increases because RNN's natural characteristic of forgetting information from long ranging data. To address this RNN's forgetting phenomenon, (Bahdanau et al., 2014) proposed an attention mechanism, however, we found that it still showed a limitation when we consider very large sequential length data such as 162 steps average in the Ubuntu Dialogue Corpus dataset (see Table 1). To overcome this limitation, we designed the HRDE architecture. The HRDE model divides long sequential text data into small chunk such as sentences, and encodes the whole text from word-level to chunk-level by using two hierarchical level of RNN architecture.

Figure 1 shows a diagram of the HRDE model. The word-level RNN part is responsible for encoding the words sequence $\boldsymbol{w}_c = \{w_{c,1:t}\}$ in each chunk. The chunk can be sentences in paragraph, paragraphs in essay, turns in dialogue or any kinds of smaller meaningful sub-set from the text. Then the final hidden states of each chunk will be fed into chunk-level RNN with its original sequence order kept. Therefore the chunk-level RNN can deal with pre-encoded chunk data with less sequential steps. The hidden states of the hierarchical RNNs are as follows:

$$
\begin{aligned}
h_{c,t} &= f_\theta(h_{c,t-1}, w_{c,t}), \\
u_c &= g_\theta(u_{c-1}, h_c),
\end{aligned}
\tag{3}
$$

where f_θ and g_θ are the RNN function in hierarchical architecture with weight parameters θ, $h_{c,t}$ is word-level RNN's hidden status at t-th word in c-th chunk. The $w_{c,t}$ is t-th word in c-th chunk of target question or answer text. The u_c is chunk-level RNN's hidden state at c-th chunk sequence, and h_c is word-level RNN's last hidden state of each chunk $h_c \in \{h_{1:c,t}\}$.

We use the same training objective as the RDE model, and the final matching probability between question and answer text is calculated using chunk-level RNN as follows:

$$
p(\text{label}) = \sigma((u_{c_q}^Q)^T M\ u_{c_a}^A + b), \tag{4}
$$

where $u_{c_q}^Q$ and $u_{c_a}^A$ are chunk-level RNN's last hidden state of each question and answer text with the dimensionality $u_c \in \mathbb{R}^{d^u}$, which involves the $M \in \mathbb{R}^{d^u \times d^u}$.

3.3 Latent Topic Clustering (LTC)

To learn how to rank QA pairs, a neural network should be trained to find the proper feature that represents the information within the data and fits the model parameter that can approximate the true-hypothesis. For this type of problem, we propose the LTC module for grouping the target data to help the neural network find the true-hypothesis with more information from the topic cluster in end-to-end training.

The blue-dotted box on the right-side of Figure 2 shows LTC structure diagram. To assign topic information, we build internal latent topic memory

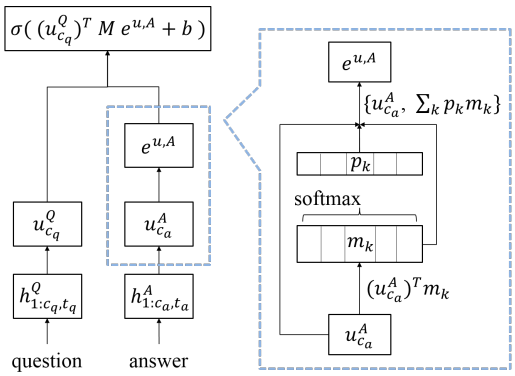

Figure 2: Diagram of the HRDE-LTC. Input vector is compared to each latent topic memory m_k to calculate cluster-info contained vector. This vector will be concatenated to original input vector.

$m \in \mathbb{R}^{d^m \times K}$, which is only model parameter to be learned, where d^m is vector dimension of each latent topic and K is number of latent topic cluster. For a given input sequence $\boldsymbol{x} = \{x_{1:t}\}$with these K vectors, we construct LTC process as follows:

$$
\begin{aligned}
p_k &= \text{softmax}((\boldsymbol{x})^T m_k), \\
\boldsymbol{x}_K &= \sum_{k=1}^{K} p_k m_k, \\
\boldsymbol{e} &= \text{concat}\{\boldsymbol{x}, \boldsymbol{x}_K\}.
\end{aligned}
\tag{5}
$$

First, the similarity between the $\boldsymbol{x}$ and each latent topic vector is calculated by dot-product. Then the resulting K values are normalized by the softmax function $\text{softmax}(z_k) = e^{z_k} / \sum_i e^{z_i}$ to produce a similarity probability p_k. After calculating the latent topic probability p_k, x_K is retrieved from summing over m_k weighted by the p_k. Then we concatenate this result with the original encoding vector to generate the final encoding vector $\boldsymbol{e}$ with the LTC information added.

Note that the input sequence of the LTC could be any type of neural network based encoding function $\boldsymbol{x} = f^{\text{enc}}_\theta(\boldsymbol{w})$ such as RNN, CNN and multilayer perceptron model (MLP). In addition, if the dimension size of $\boldsymbol{x}$ is different from that of memory vector, additional output projection layer should be placed after $\boldsymbol{x}$ before applying dot-product to the memory.

3.4 Combined Model of (H)RDE and LTC

As the LTC module extracts additional topic cluster information from the input data, we can combine this module with any neural network in their end-to-end training flow. In our experiments, we combine the LTC module with the RDE and HRDE models.

3.4.1 RDE with LTC

The RDE model encodes question and answer texts to $h^Q_{t_q}$ and $h^A_{t_a}$, respectively. Hence, the LTC module could take these vectors as the input to generate latent topic cluster information added vector $\boldsymbol{e}$. With this vector, we calculate the affinity among question and answer texts as well as additional cluster information. The following equation shows our RDE-LTC process:

$$
p(\text{label}) = \sigma((h^Q_{t_q})^T M\, e^A + b). \tag{6}
$$

In this case, we applied the LTC module only for the answer side, assuming that the answer text is longer than the question. Thus, it needs to be clustered. To train the network, we use the same training objective, to minimize cross-entropy loss, as in equation (2).

3.4.2 HRDE with LTC

The LTC can be combined with the HRDE model, in the same way it is applied to the RDE-LTC model by modifying equation (6 as follows:

$$
p(\text{label}) = \sigma((u^Q_{c_q})^T M\, e^{u,A} + b), \tag{7}
$$

where $u^Q_{c_q}$ is the final network hidden state vector of the chunk-level RNN for a question input sequence. The $e^{u,A}$ is the LTC information added vector from equation (5), where the LTC module takes the input $\boldsymbol{x} = \boldsymbol{u}^A$ from the HRDE model equation (3). The HRDE-LTC model also use the same training objective, minimizing cross-entropy loss, as in equation (2). Figure 2 shows a diagram of the combined model with the HRDE and the LTC.

4 Experimental Setup and Dataset

4.1 The Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus has been developed by expanding and preprocessing the Ubuntu Chat Logs[1], which refer to a collection of logs from the Ubuntu-related chat room for solving problem in using the Ubuntu system by (Lowe et al., 2015).

Among the utterances in the dialogues, they consider each utterance, starting from the third one, as a potential {response} while the previous utterance is considered as a {context}. The data

[1]These logs are available from http://irclogs.ubuntu.com

Dataset	# Samples			Message (Avg.)			Response (Avg.)		
	Train	Val.	Test	# tokens	# groups	# tokens /group	# tokens	# groups	# tokens /group
Ubuntu-v1	1M	35,609	35,517	162.47 ±132.47	8.43 ±6.32	20.14 ±18.41	14.44 ±13.93	1	-
Ubuntu-v2	1M	19,560	18,920	85.92 ±74.71	4.95 ±2.98	20.73 ±20.19	17.01 ±16.41	1	-
Samsung QA	163,616	10,000	10,000	12.84 ±6.42	1	-	173.48 ±192.12	6.09 ±5.58	29.28 ±31.91

Table 1: Properties of the Ubuntu and Samsung QA dataset. The message and response are {context}, {response} in Ubuntu and {question}, {answer} in the Samsung QA dataset.

was processed extracting ({context}, {response}, flag) tuples from the dialogues.

We called this original Ubuntu dataset as Ubuntu-v1 dataset. After releasing the Ubuntu-v1 dataset, researchers published v2 version of this dataset. Main updates are separating train/valid/test dataset by time so that mimics real life implementation, where we are training a model on past data to predict future data, changing sampling procedure to increase average turns in the {context}. We consider this Ubuntu dataset is one of the best dataset in terms of its quality, quantity and availability for evaluating the performance of the text ranking model.

To encode the text with the HRDE and HRDE-LTC model, a text needs to be divided into several chunk sequences with predefined criteria. For the Ubuntu-v1 dataset case, we divide the {context} part by splitting with end-of-sentence delimiter "_eos_", and we do not split the {response} part since it is normally short and does not contain "_eos_" information. For the Ubuntu-v2 dataset case, we split the {context} part in the same way as we do in the Ubuntu-v1 dataset while only using end-of-turn delimiter "_eot_". Table 1 shows properties of the Ubuntu dataset.

Question
how do i set a timer of clock in applications and development for samsung galaxy s4 mini?

Answer
1 from within the clock application, tap timer tab. 2 tap the hours, minutes, or seconds field and use the on-screen keypad to enter the hour, minute, or seconds. the timer plays an alarm at the end of the countdown. 3 tap start to start the timer. 4 tap stop to stop the timer or reset to reset the timer and start over. 5 tap restart to resume the timer counter.

Table 2: Example of the Samsung QA dataset.

4.2 Consumer Product QA Corpus

To test the robustness of the proposed model, we introduce an additional question and answer pair dataset related to an actual user's interaction with the consumer electronic product domain. We crawled data from various sources like the Samsung Electronics' official web site[2] and crowd QA web sites[3][4] in a similar way that (Yoon et al., 2016) did in building QA system for consumer products. On the official web page, we can retrieve data consisting of user questions and matched answers like frequently asked questions and troubleshooting. From the crowd QA sites, there are many answers from various users for each question. Among these answers, we choose answers from company certificated users to keep the reliability of the answers high. If there are no such answers, we skip that question answer pair. Table 2 shows an example of question-answer pair crawled from the web page. In addition, we crawl hierarchical product category information related to QA pairs. In particular, *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance* as top-level categories, and specific categories like *galaxy s7*, *tablet*, *led tv*, and *others* are used. We collected these meta-information for further use. The total size of the Samsung QA data is over 100,000 pairs and we split the data into approximately 80,000/10,000/10,000 samples to create train/valid/test sets, respectively. To create the train set, we use a QA pair sample as a ground-truth and perform negative sampling for answers among training sets to create *false*-label datasets. In this way, we generated ({question}, {answer}, flag) triples (see Table 1). We do the same procedure to create valid and test sets by only differentiating more negative sampling within each dataset to generate 9 *false*-label samples with one

[2]http://www.samsung.com/us
[3]http://answers.yahoo.com
[4]http://answers.us.samsung.com

ground-truth sample. We apply the same method in such a way that the Ubuntu dataset is generated from the Ubuntu Dialogue Corpus to maintain the consistency. The Samsung QA dataset is available via web repository. We refer the readers to Appendix A for more examples of each dataset.

4.3 Implementation Details

4.3.1 Ubuntu dataset case

To implement the RDE model, we use two single layer Gated Recurrent Unit (GRU) (Chung et al., 2014) with 300 hidden units . Each GRU is used to encode {context} and {response}, respectively. The weight for the two GRU are shared. The hidden units weight matrix of the GRU are initialized using orthogonal weights (Saxe et al., 2013), while input embedding weight matrix is initialized using a pre-trained embedding vector, the Glove (Pennington et al., 2014), with 300 dimension. The vocabulary size is 144,953 and 183,045 for the Ubuntu-v1/v2 case, respectively. We use the Adam optimizer (Kingma and Ba, 2014), with gradients clipped with norm value 1. The maximum time step for calculating gradient of the RNN is determined according to the input data statistics in Table 1.

For the HRDE model, we use two single layer GRU with 300 hidden units for word-level RNN part, and another two single layer GRU with 300 hidden units for chunk-level RNN part. The weight of the GRU is shared within the same hierarchical part, word-level and chunk-level. The other settings are the same with the RDE model case. As for the combined model with the (H)RDE and the LTC, we choose the latent topic memory dimensions as 256 in both ubuntu-v1 and ubuntu-v2. The number of the cluster in LTC module is decided to 3 for both the RDE-LTC and the HRDE-LTC cases. In HRDE-LTC case, we applied LTC module to the {context} part because we think it is longer having enough information to be clustered with. All of these hyper-parameters are selected from additional parameter searching experiments.

The dropout (Srivastava et al., 2014) is applied for the purpose of regularization with the ratio of: 0.2 for the RNN in the RDE and the RDE-LTC, 0.3 for the word-level RNN part in the HRDE and the HRDE-LTC, 0.8 for the latent topic memory in the RDE-LTC and the HRDE-LTC.

We need to mention that our implementation of the RDE module has the same architecture as the LSTM model (Kadlec et al., 2015) in ubuntu-v1/v2 experiments case. It is also the same architecture with the RNN model (Baudiš et al., 2016) in ubuntu-v2 experiment case. We implement the same model ourselves, because we need a baseline model to compare with other proposed models such as the RDE-LTC, HRDE and HRDE-LTC.

4.3.2 Samsung QA dataset case

To test the Samsung QA dataset, we use the same implementation of the model (RDE, RDE-LTC, HRDE and HRDE-LTC) used in testing the Ubuntu dataset. Only the differences are, we use 100 hidden units for the RDE and the RDE-LTC, 300 hidden units for the HRDE and 200 hidden units for the HRDE-LTC, and the vocabulary size of 28,848. As for the combined model with the (H)RDE and LTC, the dimensions of the latent topic memory is 64 and the number of latent cluster is 4. We chose best performing hyper-parameter of each model by additional extensive hyper-parameter search experiments.

All of the code developed for the empirical results are available via web repository [5].

5 Empirical Results

5.1 Evaluation Metrics

We regards all the tasks as selecting the best answer among text candidates for the given question. Following the previous work (Lowe et al., 2015), we report model performance as recall at k (R@k) relevant texts among given 2 or 10 candidates (e.g., 1 in 2 R@1). Though this metric is useful for ranking task, R@1 metric is also meaningful for classifying the best relevant text.

Each model we implement is trained multiple times (10 and 15 times for Ubuntu and the Samsung QA datasets in our experiments, respectively) with random weight initialization, which largely influences performance of neural network model. Hence we report model performance as mean and standard derivation values (Mean±Std).

5.2 Performance Evaluation

5.2.1 Comparison with other methods

As Table 3 shows, our proposed HRDE and HRDE-LTC models achieve the best performance for the Ubuntu-v1 dataset. We also find that the RDE-LTC model shows improvements from the baseline model, RDE.

[5] http://github.com/david-yoon/QA_HRDE_LTC

Model	Ubuntu-v1			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
TF-IDF [1]	0.659	0.410	0.545	0.708
CNN [2]	0.848	0.549	0.684	0.896
LSTM [2]	0.901	0.638	0.784	0.949
CompAgg [3]	0.884	0.631	0.753	0.927
BiMPM [4]	0.897	0.665	0.786	0.938
RDE	0.898 ±0.002	0.643 ±0.009	0.784 ±0.007	0.945 ±0.002
RDE-LTC	0.903 ±0.001	0.656 ±0.003	0.794 ±0.003	0.948 ±0.001
HRDE	0.915 ±0.001	0.681 ±0.001	0.820 ±0.001	0.959 ±0.001
HRDE-LTC	**0.916** ±0.001	**0.684** ±0.001	**0.822** ±0.001	**0.960** ±0.001

Table 3: Model performance results for the Ubuntu-v1 dataset. Models [1-4] are from (Lowe et al., 2015; Kadlec et al., 2015; Wang and Jiang, 2016; Wang et al., 2017), respectively.

Model	Ubuntu-v2			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
LSTM [1]	0.869	0.552	0.721	0.924
RNN [5]	0.907 ±0.002	0.664 ±0.004	0.799 ±0.004	0.951 ±0.001
CNN [5]	0.863 ±0.003	0.587 ±0.004	0.721 ±0.005	0.907 ±0.003
RNN-CNN [5]	0.911 ±0.001	**0.672** ±0.002	0.809 ±0.002	0.956 ±0.001
Attention [6] (RNN-CNN)	0.903 ±0.002	0.653 ±0.005	0.788 ±0.005	0.945 ±0.002
CompAgg [3]	0.895	0.641	0.776	0.937
BiMPM [4]	0.877	0.611	0.747	0.921
RDE	0.894 ±0.002	0.610 ±0.008	0.776 ±0.006	0.947 ±0.002
RDE-LTC	0.899 ±0.002	0.625 ±0.004	0.788 ±0.004	0.951 ±0.001
HRDE	0.914 ±0.001	0.649 ±0.001	0.813 ±0.001	0.964 ±0.001
HRDE-LTC	**0.915** ±0.002	0.652 ±0.003	**0.815** ±0.001	**0.966** ±0.001

Table 4: Model performance results for the Ubuntu-v2 dataset. Models [1,3-6] are from (Lowe et al., 2015; Wang and Jiang, 2016; Wang et al., 2017; Baudiš et al., 2016; Tan et al., 2015), respectively.

For the ubuntu-v2 dataset case, Table 4 reveals that the HRDE-LTC model is best for three cases (1 in 2 R@1, 1 in 10 R@2 and 1 in 10 R@5). Comparing the same model with our implementation (RDE) and (Baudiš et al., 2016)'s implementation (RNN), there is a large gap in the accuracy (0.610 and 0.664 of 1 in 10 R@1 for RDE and RNN, receptively). We think this is largely influenced by the data preprocessing method, because the only differences between these models is the data preprocessing, which is (Baudiš et al., 2016)'s contribution to the research. We are certain that our model performs better with the exquisite datasets which adapts extensive preprocessing method, because we see improvements from the RDE model to the HRDE model and additional improvements with the LTC module in all test cases (the Ubuntu-v1/v2 and the Samsung QA).

Model	Samsung QA			
	1 in 2 R@1	1 in 10 R@1	1 in 10 R@2	1 in 10 R@5
TF-IDF	0.939	0.834	0.897	0.953
RDE	0.978 ±0.002	0.869 ±0.009	0.966 ±0.003	0.997 ±0.001
RDE-LTC	0.981 ±0.002	0.880 ±0.009	0.970 ±0.003	0.997 ±0.001
HRDE	0.981 ±0.002	0.885 ±0.011	0.971 ±0.004	0.997 ±0.001
HRDE-LTC	**0.983** ±0.002	**0.890** ±0.010	**0.972** ±0.003	**0.998** ±0.001

Table 5: Model performance results for the Samsung QA dataset.

# clusters	Accuracy (1 in 10 R@1)		
	Ubuntu-v1	Ubuntu-v2	Samsung QA
1	0.643 ±0.009	0.610 ±0.008	0.869 ±0.009
2	0.655 ±0.005	0.616 ±0.006	0.876 ±0.011
3	**0.656** ±0.003	**0.625** ±0.004	0.877 ±0.010
4	0.651 ±0.005	0.622 ±0.005	**0.880** ±0.009

Table 6: The RDE-LTC model results with different numbers of latent clusters. "Cluster 1" is the baseline model, RDE.

In the Samsung QA case, Table 5 indicates that the proposed RDE-LTC, HRDE, and the HRDE-LTC model show performance improvements when compared to the baseline model, TF-IDF and RDE. The average accuracy statistics are higher in the Samsung QA case when compared to the Ubuntu case. We think this is due to in the smaller vocabulary size and context variety. The Samsung QA dataset deals with narrower topics than in the Ubuntu dataset case. We are certain that our proposed model shows robustness in several datasets and different vocabulary size environments.

5.2.2 Degradation Comparison for Longer Texts

To verify the HRDE model's ability compared to the baseline model RDE, we split the testset of the Ubuntu-v1/v2 datasets based on the "number of chunks" in the {context}. Then, we measured the top-1 recall (same case as 1 in 10 R@1 in Table 3, and 4) for each group. Figure 3 demonstrates that the HRDE models, in darker blue and red colors, shows better performance than the RDE models, in lighter colors, for every "number of chunks" evaluations. In particular, the HRDE models are consistent when the "number-of-chunks" increased, while the RDE models degrade as the "number-of-chunks" increased.

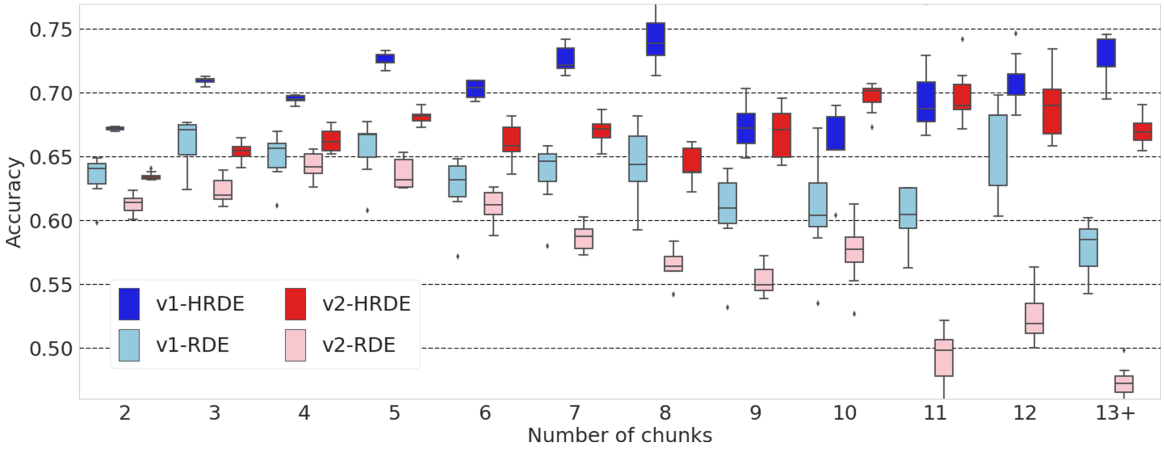

Figure 3: The HRDE and RDE model performance comparisons for the number-of-chunk in the Ubuntu dataset. Each boxplot shows average accuracy with standard deviation. The HRDE models, in darker blue and red colors, show consistent performances as the number-of-chunks increased. Meanwhile, the RDE models in lighter colors show performance degradation as the number-of-chunks increased. Furthermore, 13+ indicates all data over 13-chunks.

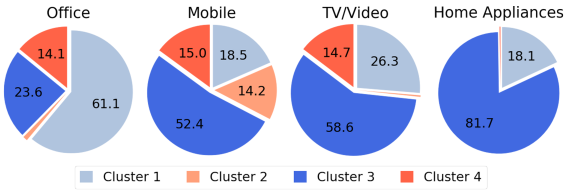

Figure 4: Examples of the cluster proportions for four real *categories* from $20k$ evaluated samples. Each color corresponds to each cluster.

Cluster	Example
1	How to adjust the brightness on the s**d300 series monitors
2	How do I reject an incoming call on my Samsung Galaxy Note 3?
3	How should I clean and maintain the microwave?
4	How do I connnect my surround sound to this TV and what type of cables do I need

Table 7: Example sentences for each cluster.

5.2.3 Effects of the LTC Numbers

We analyze the RDE-LTC model for different numbers of latent clusters. Table 6 indicates that the model performances increase as the number of latent clusters increase (until 3 for the Ubuntu and 4 for the Samsung QA case). This is probably a major reason for the different number of subjects in each dataset. The Samsung QA dataset has an internal *category* related to the type of consumer electronic products (6 top-level *categories*; *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance*), so that the LTC module makes clusters these *categories*. The Ubuntu dataset, however, has diverse contents related to issues in using the Ubuntu system. Thus, the LTC module has fewer clusters with the sparse topic compared to the Samsung QA dataset.

5.2.4 Comprehensive Analysis of LTC

We conduct quantitative and qualitative analysis on the HRDE-LTC model for four latent topic clusters. The Samsung QA dataset has *category* information; hence, latent topic clustering results can be compared with real *categories*. We randomly choose $20k$ samples containing real *category* information and evaluate each sample with the HRDE-LTC model. The cluster with the highest similarity among the latent topic clusters is considered a representative cluster of each sample.

Figure 4 shows proportion of four latent clusters among these samples according to real *category* information. Even though the HRDE-LTC model is trained without any ground-truth *category* labels, we observed that the latent cluster is formed accordingly. For instance, cluster 2 is shown mostly in "Mobile" *category* samples while "clusters 2 and 4" are rarely shown in "Home Appliance" *category* samples.

Additionally, we explore sentences with higher similarity score from the HRDE-LTC module for each four cluster. As can be seen in Table 7, "cluster 1" contains "screen" related sentences (e.g., brightness, pixel, display type) while "cluster 2" contains sentences with exclusive information re-

lated to the "Mobile" *category* (e.g., call rejection, voice level). This qualitative analysis explains why "cluster 2" is shown mostly in the "Mobile" *category* in Figure 4. We also discover that "cluster 3" has the largest portion of samples. As "cluster 3" contains "security" and "maintenance" related sentences (e.g., password, security, log-on, maintain), we assume that this is one of the frequently asked issues across all *categories* in the Samsung QA dataset. Table 7 shows example sentences with high scores from each cluster.

6 Conclusion

In this paper, we proposed the HRDE model and LTC module. HRDE showed higher performances in ranking answer candidates and less performance degradations when dealing with longer texts compared to conventional models. The LTC module provided additional performance improvements when combined with both RDE and HRDE models, as it added latent topic cluster information according to dataset properties. With this proposed model, we achieved state-of-the-art performances in Ubuntu datasets. We also evaluated our model in real world question answering dataset, Samsung QA. This demonstrated the robustness of the proposed model with the best results.

Acknowledgments

K. Jung is with the Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul, Korea. This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016M3C4A7952587), the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Industrial Technology Innovation Program (No.10073144).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127* .

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 165–180.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. 2012. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development* 56(3.4):6–1.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

James Fan, Aditya Kalyanpur, David C Gondek, and David A Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development* 56(3.4):5–1.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 285.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*. volume 242, pages 133–142.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. pages 3776–3784.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 553–562.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747* .

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, pages 4144–4150.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics* 4(1):259–272.

Seunghyun Yoon, Mohan Sundar, Abhishek Gupta, and Kyomin Jung. 2016. Automatic question answering system for consumer products. In *Proceedings of SAI Intelligent Systems Conference*. Springer, pages 1012–1016.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .

A More examples of the dataset

A.1 Ubuntu dataset

Question
"what will happend if i unmounted the ubuntu partition", "it will unmount , unless it is in use", "srr i did n't got it"

Answer	*flag*
"you cannot unmount a partition if it is currently in use"	1
"why do you not have a backup if the data is important ?"	0

Table 8: Example of the Ubuntu-v2 dataset.

A.2 Samsung QA dataset

Question
how can i place the current call on hold at any point during a conversation ?

Answer	*flag*
you can place the current call on hold at any point during a conversation . you can also make another call while you have a call in progress if your network supports this service . 1 while on a call , tap hold . this action places the current caller on hold . 2 you can later reactivate this call by tapping unhold .	1
please try to do a soft reset . turn of the phone , remove and put the battery back after 1-2 minutes . we also recommend you to clear the data of the samsung keyboard . 1 from the home screen , touch application 2 select settings 3 select application manager 4 touch the all tab 5 select samsung keyboard 6 tap on clear data .	0

Table 9: Example of the Samsung QA dataset.