ISSN 2007-9737

Unsupervised Sentence Embeddings for Answer Summarization in Non-factoid CQA

Thi-Thanh Ha[1,2], Thanh-Chinh Nguyen[1], Kiem-Hieu Nguyen[1], Van-Chung Vu[1], Kim-Anh Nguyen[1]

[1] Ha Noi University of Science and Technology, VietNam

[2] Thai Nguyen University of Information and Communication Technology, VietNam

htthanh@ictu.edu.vn

Abstract. This paper presents a method for summarizing answers in Community Question Answering. We explore deep Auto-encoder and Long-short-term-memory Auto-encoder for sentence representation. The sentence representations are used to measure similarity in Maximal Marginal Relevance algorithm for extractive summarization. Experimental results on a benchmark dataset show that our unsupervised method achieves state-of-the-art performance while requiring no annotated data.

Keywords. Summarizing answers, non-factoid questions, multi-document summarization, community question-answering, auto encoder, LSTM.

1 Introduction

In Community Question and Answering (CQA) services (Yahoo Answers[1], StackOverflow[2]), users can post new questions and answer existing questions. Four main problems in CQA are [10]: (1) finding similar questions given a new question, (2) finding answers given a new question, (3) measuring answer quality and its effect on question retrieval, and (4) finding experts in a community. Our task of summarizing answers posits in the third problem.

Among the answers, question owner selects one or several ones as best answer(s). 48% questions have a unique answer [10]. Best answers could be incomplete, particularly for complex questions or non-factoid questions (against factoid questions which requires concise facts). This raises the need for answer summarization in CQA. Researchers have been using text summarization techniques for summarizing factoid, non-factoid, as well as multi-sentence and complex questions [19, 17, 2].

This work focuses on using unsupervised sentence representation to tackle answer summarization in non-factoid CQA. Two neural models including deep Auto-Encoder (AE) and Long-short-term-memory Auto-Encoder (LSTM-AE) [5, 8] are explored to capture semantic and syntactic information and generate low-dimensional vectors, which are later used for measuring sentence similarity.

We aim at tackling three main challenges: sparsity, diversity, and genre adaptation. Neural embeddings help overcome sparsity of short texts (i.e. questions and answer sentences in this work). The Maximal Marginal Relevance (MMR) algorithm [1] balances question relevance and summary diversity. Last but not least, representations based on Yahoo-Webscope are expected to be more suitable for CQA.

The rest of the paper is organized as follows. Related works are discussed in Section 2. Section 3 is dedicated to our method for answer summarization. Experiments are presented in Section 4. Finally, Section 5 concludes the paper.

[1] https://answers.yahoo.com/
[2] https://stackoverflow.com/

2 Related Work

Techniques in text summarization have been applied to answer summarization in question-answering [19]. Liu applied clustering on open questions and opinion questions [10]. Tomasoni exploited metadata and proposed concept scoring functions based on semantic overlap [18]. Other approaches aimed at solving the optimization problem for selecting a subset of sentences that maximizes an objective function under length constraint.

Integer linear programming was successfully applied to summarize answers in CQA [18]. Chan proposed using Conditional Random Fields to deal with the *incomplete answer* problem and complex multi-sentence questions. The author showed a systematic way to model semantic contextual interactions between answer sentences, based on question segmentation; Both textual and non-textual features were explored [2].

Researchers have been developing techniques to learn neural text embeddings [5, 11, 7, 4, 13, 16]. Auto-encoder was applied to query-oriented single-document summarization [20]. In another direction, sequence-to-sequence architecture was applied to abstractive summarization [12, 15, 3]. The most related works to ours on answer summarization in non-factoid CQA were presented in [14, 17], using sentence vectors generated from Paragraph Vector [7] and Convolutional Neural Network (CNN), in that order.

3 Sentence Embeddings for Answer Summarization

The proposed answer summarization framework is demonstrated in Fig 1. Given a pair of question q and its answers $\{A_i\}$, answer sentences are first extracted to generate of a set of sentences $\{S_i\}$. The sentence representation block uses Yahoo-Webscope to learn models and to generate low-dimensional vectors q' and $\{x_i\}$ for q and $\{S_i\}$, respectively. MMR algorithm takes q' and $\{x_i\}$ as inputs and generates an answer summary.

3.1 Sentence Representation

Neural networks are effective in representing semantic and syntactic information of sentences in low-dimensional vectors. This paper investigates two unsupervised neural models, i.e. Deep Auto-Encoder and Long Short-Term Memory (LSTM) Auto-Encoder [8] for sentence representation.

3.1.1 Deep Auto-Encoder

An Auto-Encoder neural network is a generative model that aims at reconstructing its own inputs. Our deep Auto-Encoder model is introduced in Fig 2. It has four encoding layers:

$$h_1 = \sigma(W_1.X), \tag{1}$$

$$h_2 = \sigma(W_2.h_1), \tag{2}$$

$$h_3 = \sigma(W_3.h_2), \tag{3}$$

$$h = \sigma(W_4.h_3). \tag{4}$$

A sentence X is put into the network with tf-idf weights. X is very sparse because it only contains a small number of words while its dimension is the size of vocabulary. The Auto-Encoder can learn a distributed semantic representation with low dimension. The layer h is used for sentence representation. Decoding layers are:

$$h_3^{'} = \sigma(W_4^{'}.h), \tag{5}$$

$$h_2^{'} = \sigma(W_3^{'}.h_3^{'}), \tag{6}$$

$$h_1^{'} = \sigma(W_2^{'}.h_2^{'}), \tag{7}$$

$$X^{'} = \sigma(W_1^{'}.h_1^{'}), \tag{8}$$

where sigmoid function is:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{9}$$

The squared error loss is:

$$J(X, X^{'}) = \|X - X^{'}\| = \sum_V (X_i - X_i^{'})^2, \tag{10}$$

where V is vocabulary size.

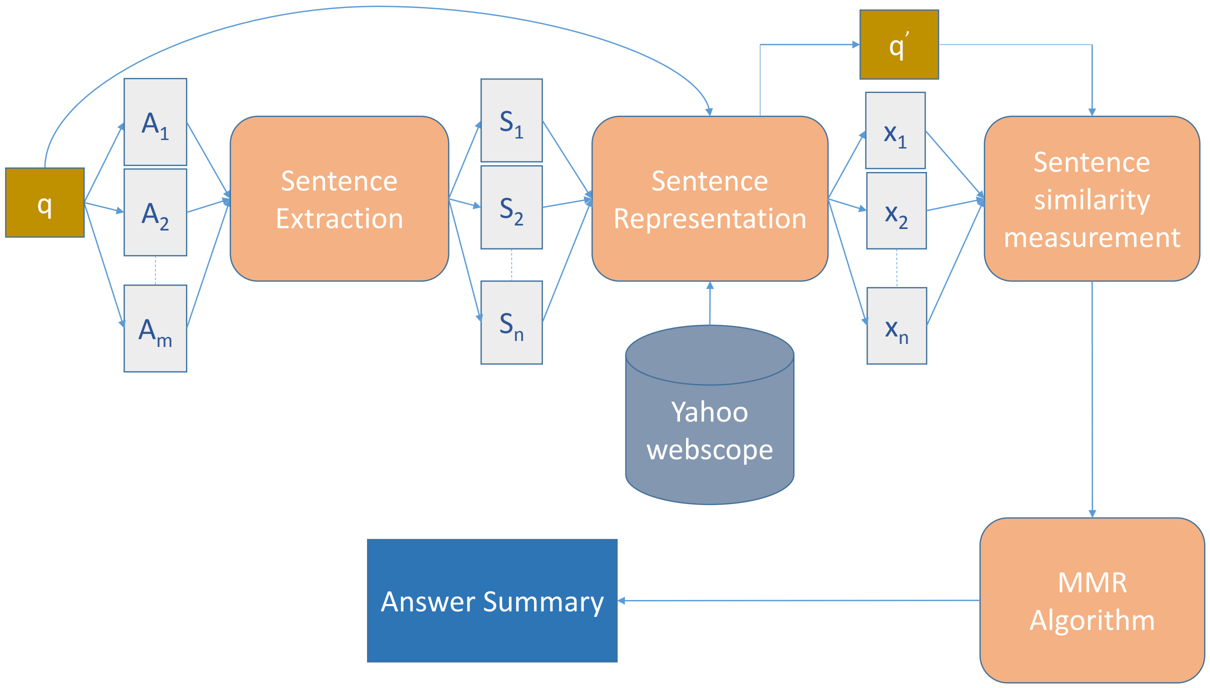

Fig. 1. Framework for answer summarization in non-factoid CQA

3.1.2 LSTM Auto-Encoder

Deep Auto-Encoder doesn't capture syntactic information in word order. We propose using LSTM Auto-encoder (Fig 3), which was first introduced in [8]. This model learns sentence in an unsupervised manner and captures both syntactic information in word order and semantic information in word embeddings:

$$h_t(enc) = LSTM^{word}_{encode}(e_t, h_{t-1}(enc)), \quad (11)$$

h_{ends} is used to present the input sentence

$$e^s = h_{ends}, \quad (12)$$

$$h_t(dec) = LSTM_{decode}(e_t, h_{t-1}(dec)). \quad (13)$$

The decoder sequentially predicts sentence words using a softmax function:

$$P(x'_t|\Delta) = softmax(e_{t-1}, h_{t-1}(dec)), \quad (14)$$

e_t is an embedding for word at position t and generated by the $LSTM_{decode}$. The encoder and decoder use two different LSTMs with two different sets of parameters.

Our loss function:

$$J(X, X') = 1/N \sum_{i<N} H(e_i, e'_i), \quad (15)$$

where H is the Cross-entropy error function. The LSTM model at time t is defined as follows:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W. \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix}, \quad (16)$$

$$c_t = f_t.c_{t-1} + i_t.l_t, \quad (17)$$

$$h_t = o_t.c_t. \quad (18)$$

3.2 Extractive Summarization

MMR is applied to generative extractive summaries (Algorithm 1). It is a greedy algorithm which incrementally selects a sentence by maximizing a

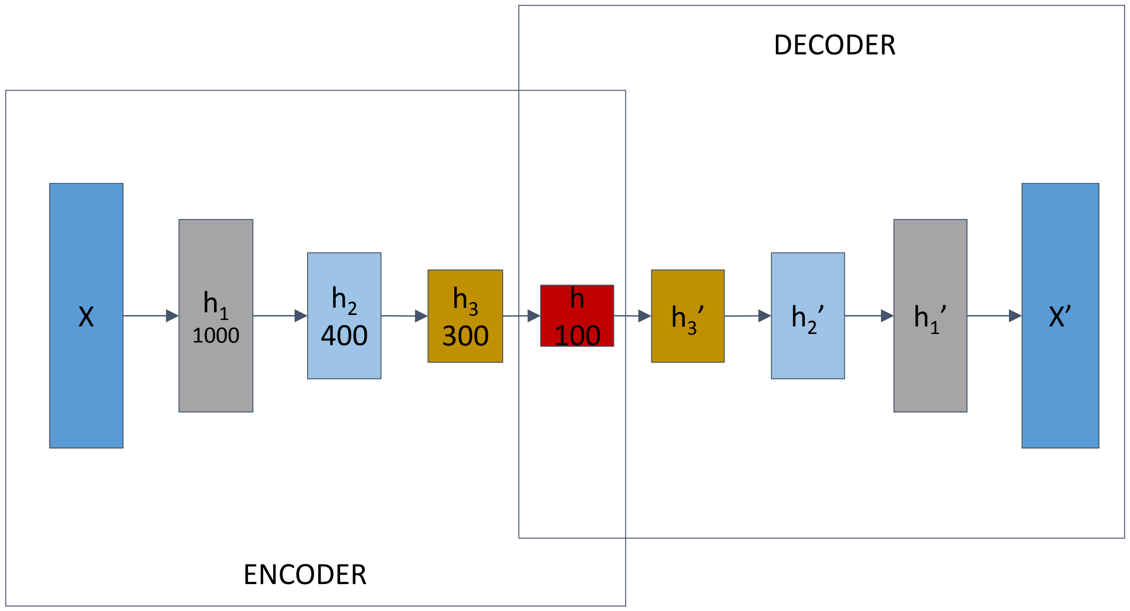

Fig. 2. Deep Auto-Encoder: h (the red block) is used for sentence representation

linear combination of query relevance and summary diversity (line 3). Here the hyper-parameter κ takes a value in $[0, 1]$. $Sim(s, q)$ and $sim(s, s')$ are sentence similarity. q is the question. S is the set of all sentences in the answers. L is the limit length of a summary. R is the set of summary sentences.

Algorithm 1 Maximal marginal relevance (MMR)

```
Input: $q, S, L$
Output: $R$
Initialize: $R$=$\varnothing$; Ranked list of summary sentences;
1: repeat
2:     Find a sentence $s$ by MMR with parameter $0 \leq \kappa \leq 1$, so that
3:     $s = \arg\max_{s \in S/R}(\kappa.sim(s, q) - (1 - \kappa).\max_{s' \in R} sim(s, s')$
4:     $R$=$R \cup s$;
5: until | R | $> L$;
6: return $R$;
```

Sentence similarity is computed by cosine similarity:

$$sim(s_1, s_2) = \frac{s_1.s_2}{\|s_1\|.\|s_2\|}. \qquad (19)$$

4 Evaluation

4.1 Datasets

L6 - Yahoo! Answers Comprehensive Questions and Answers corpus[3] from Yahoo Webscope was used for unsupervised learning of sentence representation (Table 1).

We used the test dataset in [17] for evaluation[4]. The dataset contains manual summaries with the limited length of 250 words. In our experiments, limited summary length was selected accordingly ($L = 250$ in MMR).

[3]`https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`

[4]We have no access to train and dev datasets.

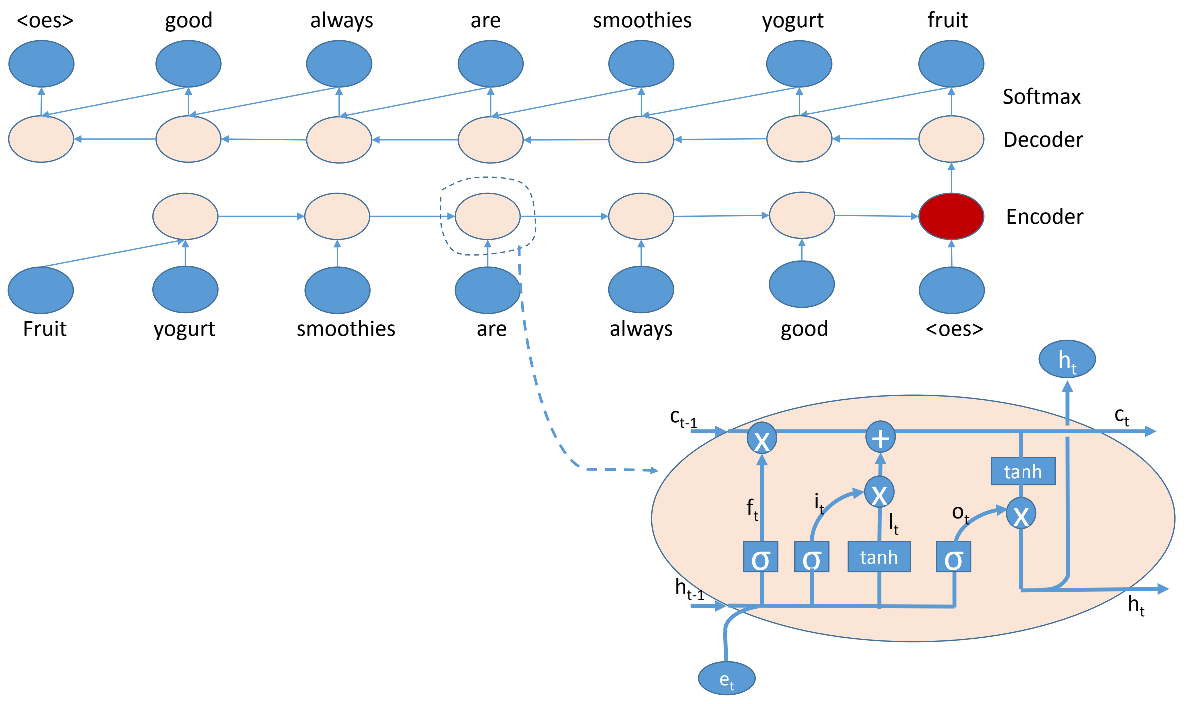

Fig. 3. Long-short-term-memory Auto-Encoder: The last encoding LSTM cell (the red node) is used for sentence representation

Table 1. Yahoo Webscope corpus

Statistics	Size
Questions	87,390
Answers	314,446
Answer sentences	1,662,497

Table 2. Test dataset

Statistics	Size
Non-factoid questions	100
Answers	361
Answer sentences	2,793
Words	59,321
Manual summaries	275
Avg. summaries per question	2.75

4.2 Experimental Setup

Each input sentence vector put into AE is represented using tf-idf. The vocabulary was created by lowercasing, removing the stopwords, rare words (below 10 times), stemming, and normalizing number. The auto-encoder has four layers for encoding, and four layers for decoding. Layer h with 100 dimensions is used to present sentence. Learning parameters for back propagation and Adam algorithm[6] were: learning rate $\eta = 0.001$; batch size = 128 sentences; 20 epochs. The model was trained on Yahoo-webscope in eight hours with a machine of 20 CPUs.

Word embeddings from word2vec[5] on Google news of size 300 were fed into LSTM-AE. When a word was not in the vocabulary of pre-trained word embeddings, its embedding was sampled from a normal distribution. Commas, colons were converted to <dot>.

[5]https://github.com/mmihaltz/word2vec

Periods, end marks were converted to <eos>. Learning parameters were: batch size of 128 sentences, 20 epochs, learning rate $\eta = 0.001$. It took three weeks with a machine of 20 CPUs to train this model on Yahoo-webscope. Both AE and LSTM-AE were implemented on Tensorflow.

4.3 Experimental Results

ROUGE metric [9] was used to evaluate text summarization. At first, the results of two baselines, tfidf and tf-idf weighted average word embeddings, are shown in Table 3. AE, LSTM-AE and a combination of AE and LSTM-AE by concatenating the two sentence embeddings (mentioned as CONCAT) are compared. The results are in Figure 4.

As we only have the test dataset, experiments with different values of κ as the only hyper-parameter (of MMR) were conducted. LSTM-AE with $\kappa = 0.3$ was selected as our representative to compare with related works. Last but not least, with $\kappa = 0.3$, linear combination of AE and LSTM-AE similarities was investigated (Table 5):

$$sim(s_1, s_2) = \alpha.sim_{AE}(s_1, s_2),$$
$$+(1-\alpha).sim_{LSTM-AE}(s_1, s_2),$$

where α is hyper-parameter.

As expected, *Word2vec* outperforms *tfidf* by large margin (Table 3) thanks to low dimensional vectors and semantic information. However, *Word2vec* is not on par with AE and LSTM-AE (Figure 4). This is because the former straightforwardly derives sentence embeddings from word embeddings by weighted average; while sentence vectors are parameters of the two latter models that need to be learned from data. With $\kappa < 0.5$, LSTM-AE beats AE on all the metrics. When $\kappa > 0.5$, AE performs better on ROUGE-1 and ROUGE-2. This is possible because a large value of κ prefer diversity to relevance. Overall, LSTM-AE is a better choice. It is worth noting that concatenating the two models doesn't bring significant improvement (Figure 4).

LSTM-AE with $\kappa = 0.3$ was compaired to state-of-the-art methods. DOC2VEC [14] uses Paragraph Vector [7] to generate sentence representation and sparse coding to detect salient sentences. However, it is not clear on which data Paragraph Vector was learned and how sentences were represented. CNN learns sentence embeddings from annotated answer sentences, i.e. sentences with labels as summary or non-summary. Relevant sentences from Wikipedia are also retrieved to overcome sparsity. Low-dimensional sentence vectors are first put into sparse coding and then MMR to generate summaries. Here, the baseline BestAns selects the best answers as summaries.

Interestingly, our unsupervised sentence representation performs slightly better than supervised one without annotated data (Table 4). LSTM-AE outperforms DOC2VEC. The reason could be two-fold: i) Paragraph Vector introduces paragraph (i.e. sentence in this case) context via so-called *paragraph_id* additional token in the input layer, and sampling several windows through the sentence. Meanwhile, LSTM-AE captures semantic and syntactic of the sentence in the last encoding LSTM cell and uses it for sentence representation. ii) LSTM-AE was trained on Yahoo-Webscope, a large corpus of questions and answers from communities.

This could make sentence representation more suitable to CQA tasks. On the other hand, we have no clue on which data Paragraph Vector is trained in DOC2VEC; and why ROUGE-2 reported in [14] is higher than both CNN and our method. In the future, we are going to reimplement DOC2VEC, with Yahoo-Webscope as training data for Paragraph Vector, to investigate in more details.

Table 5 shows that linear combination of sentence similarities is more effective than concatenating the representations of sentence pairs (Figure 4).

5 Conclusions and Discussions

The paper presents an approach to summarizing answers for non-factoid questions in CQA using unsupervised neural sentence embeddings. Semantic and syntactic information, as well as genre and domain knowledge are incorporated in

Table 3. Evaluating two baselines

	Word2Vec			Tfidf		
κ	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
0.1	0.621	0.529	0.607	0.532	0.282	0.464
0.2	0.619	0.524	0.606	0.531	0.282	0.463
0.3	0.618	0.523	0.605	0.532	0.281	0.464
0.4	0.615	0.518	0.600	0.530	0.279	0.467
0.5	0.622	0.525	0.604	0.529	0.279	0.464
0.6	0.614	0.513	0.605	0.528	0.278	0.467
0.7	0.610	0.507	0.607	0.529	0.280	0.489
0.8	0.609	0.504	0.610	0.530	0.285	0.488
0.9	0.611	0.505	0.603	0.532	0.288	0.488
1.0	0.608	0.501	0.601	0.532	0.289	0.489

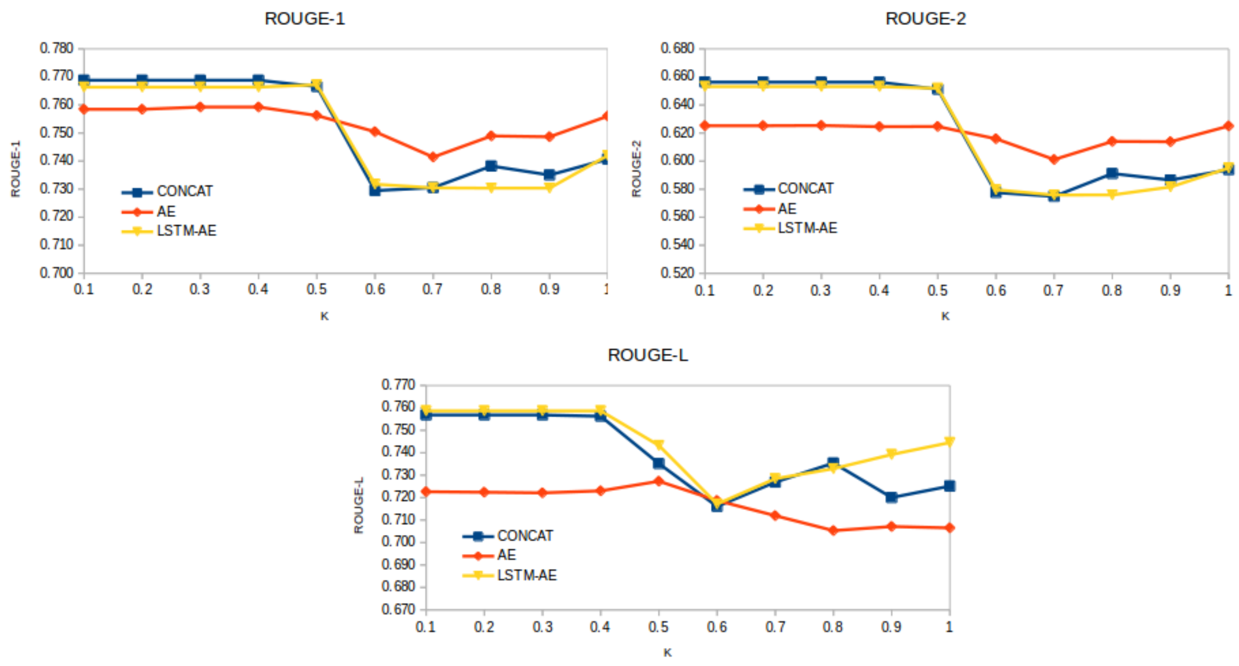

Fig. 4. Performance on varying κ in MMR

low-dimensional vectors. Empirical results demonstrated the effectiveness of these representations, particularly ones generated by LSTM-AE. Our method outperforms other methods and is on par with a method based on supervised sentence representation. In the future, we are going to apply drop-out in learning neural models, and use Restricted Boltzmann Machines to initialize Auto-Encoder to enhance their output representation. Moreover, encouraging by results on CQA answer summarization, we are going to investigate LSTM-AE on extractive text summarization and CQA problems.

Table 4. Comparison to state-of-the-art methods

Method	**Rouge-1**	**Rouge-2**	**Rouge-L**
BestAns	0.473	0.390	0.463
DOC2VEC + sparse coding	0.753	0.678	0.750
CNN + document expansion + sparse coding + MMR	0.766	0.646	0.753
LSTM-AE	0.766	0.653	0.759

Table 5. Evaluating linear combination of AE similarity and LSTM-AE similarity

α	**Rouge-1**	**Rouge-2**	**Rouge-L**
0.1	0.771	0.661	0.761
0.2	0.771	0.661	0.760
0.3	0.771	0.661	0.760
0.4	0.770	0.660	0.759
0.5	0.770	0.659	0.759
0.6	0.771	0.658	0.759
0.7	0.772	0.662	0.763
0.8	0.772	0.662	0.763
0.9	0.771	0.660	0.759

Acknowledgements

This work was partially funded by the T2018-07-03 project Managed ICTU.

References

1. **Carbonell, J. & Goldstein, J. (1998).** The use of MMR, diversity-based reranking for reordering documents and producing summaries. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, ACM, New York, NY, USA, pp. 335–336.
2. **Chan, W., Zhou, X., Wang, W., & Chua, T.-S. (2012).** Community answer summarization for multi-sentence question with group l1 regularization. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 582–591.
3. **Chopra, S., Auli, M., & Rush, A. M. (2016).** Abstractive sentence summarization with attentive recurrent neural networks. *HLT-NAACL*.
4. **Gouws, S., Bengio, Y., & Corrado, G. (2015).** Bilbowa: Fast bilingual distributed representations without word alignments. *Proceedings of the 32nd International Conference on Machine Learning*.
5. **Hinton, G. & Salakhutdinov, R. (2006).** Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, No. 5786, pp. 504 – 507.
6. **Kingma, D. P. & Ba, J. (2014).** Adam: A method for stochastic optimization. *CoRR*, Vol. abs/1412.6980.
7. **Le, Q. V. & Mikolov, T. (2014).** Distributed representations of sentences and documents. *CoRR*, Vol. abs/1405.4053.
8. **Li, J., Luong, M., & Jurafsky, D. (2015).** A hierarchical neural autoencoder for paragraphs and documents. *CoRR*, Vol. abs/1506.01057.
9. **Lin, C.-Y. (2004).** ROUGE: a package for automatic evaluation of summaries. *Text Summarization Branches Out*.
10. **Liu, Y., Li, S., Cao, Y., Lin, C.-Y., Han, D., & Yu, Y. (2008).** Understanding and summarizing answers in community-based question answering services. *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 497–504.
11. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. *CoRR*, Vol. abs/1310.4546.
12. **Nallapati, R., Xiang, B., & Zhou, B. (2016).** Sequence-to-sequence rnns for text summarization. *CoRR*, Vol. abs/1602.06023.
13. **Qiu, X. & Huang, X. (2015).** Convolutional neural tensor network architecture for community-based question answering. *Proceedings of the 24th*

International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, pp. 1305–1311.

14. **Ren, Z., Song, H., Li, P., Liang, S., Ma, J., & de Rijke, M. (2016).** Using sparse coding for answer summarization in non-factoid community question-answering.

15. **Rush, A. M., Chopra, S., & Weston, J. (2015).** A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

16. **Severyn, A. & Moschitti, A. (2016).** Modeling relational information in question-answer pairs with convolutional neural networks. *CoRR*, Vol. abs/1604.01178.

17. **Song, H., Ren, Z., Liang, S., Li, P., Ma, J., & de Rijke, M. (2017).** Summarizing answers in non-factoid community question-answering. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, ACM, New York, NY, USA, pp. 405–414.

18. **Tomasoni, M. & Huang, M. (2010).** Metadata-aware measures for answer summarization in community question answering. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 760–769.

19. **Wang, M. (2006).** A survey of answer extraction techniques in factoid question answering ravichandran, deepak, abharam ittycheriah, and salim roukos. 2003. automatic derivation of surface text patterns for a maximum entropy based question answering system. *In Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL*.

20. **Yousefiazar, M. (2015).** *Query-oriented Single-document Summarization Using Unsupervised Deep Learning*.

Article received on 20/01/2018; accepted on 05/03/2018.
Corresponding author is Thi-Thanh Ha.