Unsupervised Sentence Embeddings for Answer Summarization in Non-factoid CQA

Thi-Thanh Ha¹,², Thanh-Chinh Nguyen¹, Kiem-Hieu Nguyen¹, Van-Chung Vu¹, Kim-Anh Nguyen¹

¹ Ha Noi University of Science and Technology, VietNam
² Thai Nguyen University of Information and Communication Technology, VietNam

htthan@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 Answers¹, StackOverflow²), 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.

¹https://answers.yahoo.com/
²https://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 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:

\[
\begin{align*}
h_1 &= \sigma(W_1.X), \\
h_2 &= \sigma(W_2.h_1), \\
h_3 &= \sigma(W_3.h_2), \\
h &= \sigma(W_4.h_3).
\end{align*}
\]

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:

\[
\begin{align*}
h'_3 &= \sigma(W'_4.h), \\
h'_2 &= \sigma(W'_3.h'_3), \\
h'_1 &= \sigma(W'_2.h'_2), \\
X' &= \sigma(W'_1.h'_1),
\end{align*}
\]

where sigmoid function is:

\[
\sigma(x) = \frac{1}{1 + e^{-x}}.
\]

The squared error loss is:

\[
J(X, X') = \|X - X'\| = \sum_v (X_i - X'_i)^2,
\]

where \( V \) is vocabulary size.
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:

\[ h_t^{(enc)} = LSTM_{word}(e_t, h_{t-1}^{(enc)}) \]  \hspace{1cm} (11)

\[ h_{ends} \] is used to present the input sentence

\[ e^s = h_{ends} \] \hspace{1cm} (12)

\[ h_t^{(dec)} = LSTM_{decode}(e_t, h_{t-1}^{(dec)}) \] \hspace{1cm} (13)

The decoder sequentially predicts sentence words using a softmax function:

\[ P(x_t | \Delta) = softmax(e_{t-1}, h_{t-1}^{(dec)}) \] \hspace{1cm} (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:

\[ J(X, X') = \frac{1}{N} \sum_{i<N} H(e_i, e_i') \] \hspace{1cm} (15)

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

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

\[ c_t = f_t.c_{t-1} + i_t.l_t, \] \hspace{1cm} (17)

\[ h_t = o_t.c_t. \] \hspace{1cm} (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

Algorithm 1 Maximal marginal relevance (MMR)

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

Sentence similarity is computed by cosine similarity:
\[
\text{sim}(s_1, s_2) = \frac{s_1 \cdot s_2}{\|s_1\| \cdot \|s_2\|}.
\] (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=1
\(^4\)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

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions</td>
<td>87,390</td>
</tr>
<tr>
<td>Answers</td>
<td>314,446</td>
</tr>
<tr>
<td>Answer sentences</td>
<td>1,662,497</td>
</tr>
</tbody>
</table>

Table 2. Test dataset

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-factoid questions</td>
<td>100</td>
</tr>
<tr>
<td>Answers</td>
<td>361</td>
</tr>
<tr>
<td>Answer sentences</td>
<td>2,793</td>
</tr>
<tr>
<td>Words</td>
<td>59,321</td>
</tr>
<tr>
<td>Manual summaries</td>
<td>275</td>
</tr>
<tr>
<td>Avg. summaries per question</td>
<td>2.75</td>
</tr>
</tbody>
</table>

4.2 Experimental Setup

Each input sentence vector put into AE is represented using tf-idf. The vocabulary was created by lowering, 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 \(<\text{dot}>\).

\(^5\)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 $\kappa$ 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 $\alpha$ 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 $\kappa$ 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 compared 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

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Word2Vec</th>
<th>Tfidf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rouge-1</td>
<td>Rouge-2</td>
</tr>
<tr>
<td>0.1</td>
<td>0.621</td>
<td>0.529</td>
</tr>
<tr>
<td>0.2</td>
<td>0.619</td>
<td>0.524</td>
</tr>
<tr>
<td>0.3</td>
<td>0.618</td>
<td>0.523</td>
</tr>
<tr>
<td>0.4</td>
<td>0.615</td>
<td>0.518</td>
</tr>
<tr>
<td>0.5</td>
<td>0.622</td>
<td>0.525</td>
</tr>
<tr>
<td>0.6</td>
<td>0.614</td>
<td>0.513</td>
</tr>
<tr>
<td>0.7</td>
<td>0.610</td>
<td>0.507</td>
</tr>
<tr>
<td>0.8</td>
<td>0.609</td>
<td>0.504</td>
</tr>
<tr>
<td>0.9</td>
<td>0.611</td>
<td>0.505</td>
</tr>
<tr>
<td>1.0</td>
<td>0.608</td>
<td>0.501</td>
</tr>
</tbody>
</table>

Fig. 4. Performance on varying $\kappa$ 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestAns</td>
<td>0.473</td>
<td>0.390</td>
<td>0.463</td>
</tr>
<tr>
<td>DOC2VEC + sparse coding</td>
<td>0.753</td>
<td>0.678</td>
<td>0.750</td>
</tr>
<tr>
<td>CNN + document expansion + sparse coding + MMR</td>
<td>0.766</td>
<td>0.646</td>
<td>0.753</td>
</tr>
<tr>
<td>LSTM-AE</td>
<td>0.766</td>
<td>0.653</td>
<td>0.759</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>α</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.771</td>
<td>0.661</td>
<td>0.761</td>
</tr>
<tr>
<td>0.2</td>
<td>0.771</td>
<td>0.661</td>
<td>0.760</td>
</tr>
<tr>
<td>0.3</td>
<td>0.771</td>
<td>0.661</td>
<td>0.760</td>
</tr>
<tr>
<td>0.4</td>
<td>0.770</td>
<td>0.660</td>
<td>0.759</td>
</tr>
<tr>
<td>0.5</td>
<td>0.770</td>
<td>0.659</td>
<td>0.759</td>
</tr>
<tr>
<td>0.6</td>
<td>0.771</td>
<td>0.658</td>
<td>0.759</td>
</tr>
<tr>
<td>0.7</td>
<td>0.772</td>
<td>0.662</td>
<td>0.763</td>
</tr>
<tr>
<td>0.8</td>
<td>0.772</td>
<td>0.662</td>
<td>0.763</td>
</tr>
<tr>
<td>0.9</td>
<td>0.771</td>
<td>0.660</td>
<td>0.759</td>
</tr>
</tbody>
</table>

Acknowledgements

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

References


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