Summarizing Answers in Non-Factoid Community Question-Answering

Hongya Song†, hongya.song.sdu@gmail.com Zhaochun Ren‡, zhaochun.ren@ucl.ac.uk Shangsong Liang‡, shangsong.liang@ucl.ac.uk

Piji Li§, pjili@se.cuhk.edu.hk Jun Ma†, majun@sdu.edu.cn Maarten de Rijke††, derijke@uva.nl

†Shandong University, Jinan, China
‡University College London, London, United Kingdom
§The Chinese University of Hong Kong, Hong Kong, China
††University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

We aim at summarizing answers in community question-answering (CQA). While most previous work focuses on factoid question-answering, we focus on the non-factoid question-answering. Unlike factoid CQA, non-factoid question-answering usually requires passages as answers. The shortness, sparsity and diversity of answers form interesting challenges for summarization. To tackle these challenges, we propose a sparse coding-based summarization strategy that includes three core ingredients: short document expansion, sentence vectorization, and a sparse-coding optimization framework. Specifically, we extend each answer in a question-answering thread to a more comprehensive representation via entity linking and sentence ranking strategies. From answers extended in this manner, each sentence is represented as a feature vector trained from a short text convolutional neural network model. We then use these sentence representations to estimate the saliency of candidate sentences via a sparse-coding framework that jointly considers candidate sentences and Wikipedia sentences as reconstruction items. Given the saliency vectors for all candidate sentences, we extract sentences to generate an answer summary based on a maximal marginal relevance algorithm. Experimental results on a benchmark data collection confirm the effectiveness of our proposal method in answer summarization of non-factoid CQA, and moreover, its significant improvement compared to state-of-the-art baselines in terms of ROUGE metrics.

Keywords

Community question-answering; Sparse coding; Short text processing; Document summarization

1. INTRODUCTION

In recent years, we have witnessed a rapid growth in the number of users of community question-answering (CQA). In the wake of this development, an increasing number of approaches to CQA retrieval have been proposed, addressing a wide range of tasks, including answer ranking [47, 49, 50], answer extraction [51], multimedia QA [28], answer generation [46, 53], and question classification [9, 10]. There has been a very strong focus on factoid CQA, in which typically a single correct answer exists, e.g., “Where was X born?” In contrast, in non-factoid CQA, multiple sparse and diverse sentences may make up the answers together. However, their shortness, sparsity and diversity make it difficult to identify all of the information that together covers all aspects of a question. Document summarization is an effective way to summarize salient information and generate a relevant and diverse answer for a given input question, in particular, in the context of non-factoid question-answering [13, 43]. However, traditional document summarization methods face a number of challenges when used for summarizing non-factoid answers in CQA.

The task on which we focus in this paper is *answer summarization in non-factoid community question-answering* [43]. Unlike previous work on document summarization, answer summarization in non-factoid CQA focuses on collecting all relevant and meaningful sentences to the input question from candidate answers. Fig. 1 shows an example of an answer summary given a question about “how to cure indigestion.” We can find that the summary extracts sentences from candidate answers and covers important and diverse aspects of the answers to the question. Compared to traditional multi-document summarization task that usually focuses on summarizing news articles, summarizing answers in non-factoid CQA faces specific challenges: (1) The shortness of answers in non-factoid CQA is an obstacle for document summarization methods in answer summarization. (2) The sparsity of syntactic and context information hinders the summarization process using traditional representation of short text, based on term frequency or latent topic modeling [34]. (3) Summarization in non-factoid CQA is a recall-oriented problem, in which we need to recall as much relevant information as possible. However, the diverse topic distribution of answers in non-factoid CQA makes it difficult to generate a summary with high recall.

In this paper, we address the answer summarization problem
in non-factoid CQA by utilizing a novel sparse-coding strategy. To address the shortness, sparsity, and diversity mentioned above, we proceed as follows. To tackle the shortness problem, previous works have proved that document expansion is effective in short text processing; Inspired by these works, in this paper, we employ an alternative strategy involving entity linking [29] and sentence ranking to collect and filter relevant information from Wikipedia. To tackle the challenge of sparsity in sentence presentation, we apply short text convolutional neural networks (CNNs) to model sentences given the input question. By considering the match between each candidate sentence and things should begin to improve. Eating a great deal of red meat and fast foods is not good for the digestion of foods and moving them along the intestinal track. Start eating more foods from the fruit and vegetable group and things should begin to improve. Eating a great deal of red meat and fast foods is not good for the digestion of foods and moving them along the intestinal track. Our answer summarization result:

1. Don't get a mocha, it only works with the flavored syrups, and I'd recommend at least a double shot.
2. If it doesn't work at all you're out a $3 drink and you didn't have to stick anything up your rear end.
3. If it doesn't work at all you're out a $3 drink and you didn't have to stick anything up your rear end.
4. It is a laxative; use it with the flavored syrups and I'd recommend at least a double shot.
5. What is your experience with this medication but the lack of fiber in your diet.

Figure 1: An example answer summary for a question-answering thread about "how to cure indigestion." The answer summary extracts sentences from candidate answers.
cial media documents, e.g., tweets, blogs, and web forum posts [5, 7, 27, 32, 33]. Tweets summarization focuses on extracting a group of representative tweets out of a set of tweets [5, 35, 42]. MDS has been applied to web forums threads to select salient passages from candidate posts [33]. Previous work on blog summarization explores the effect of comments or social contexts on multi-document summarization [12, 18]. Li et al. [18] propose a reader-aware MDS paradigm that can generate compressive summaries jointly considering news documents and user comments. Answer summarization in community question-answering was first proposed by Zhou et al. [58], with a solution based on a head-modifier-relation triple representation of document content [11]. Tomasoni and Huang [43] exploit metadata to bias automatic answer summarization toward high quality information. However, all approaches so far neglect the shortness, sparsity and diversity challenges in answer summarization of non-factoid community question-answering.

2.2 Community question-answering retrieval

Research into community question-answering (CQA) is expanding due to the popularity of CQA on the web [57]. Recent research on community question-answering retrieval can be divided into factoid CQA research and non-factoid CQA research. Research on factoid community question-answering has seen a significant growth [9, 23, 24, 30, 38, 39, 53]. Feng et al. [9] present a question classifier using information distance. Several methods for answer sentence selection have been proposed with data from the TREC Question-Answering track [48, 51, 52]. Based on deep neural network architectures, several approaches have been proposed to select the best answer or question [30, 39, 53]. Severyn and Moschitti [39] propose a convolutional network for re-ranking pairs of a candidate answer and a question. Serban et al. [38] present a large question-answer pair corpus produced by applying a novel neural network architecture. Carmel et al. [3] take into account the syntactic function of the terms within CQA texts as an important factor affecting their importance for retrieval. Wang et al. [46] propose a retrieval-based automatic response model for short-text conversation. Omari et al. [30] promote CQA answers not only by their relevance to the question but also by diversity and novelty compared to other answers. Liu et al. [23] design a set function to re-rank questions given a user review. Zhao et al. [56] employ users’ social networks for inferring a user model, and thus improve the performance of expert finding in CQA. Multimedia CQA has also received attention recently [24, 28].

Several approaches have already been proposed for the non-factoid CQA [43, 50]. Most recent work on non-factoid CQA retrieval focuses on the task of passage retrieval [6, 50]. Yang et al. [50] design semantic and context features for answer sentence retrieval in non-factoid CQA. Using a learning-to-rank approach, Chen et al. [6] find that searching relevant answer sentences can be leveraged in a feature-based framework by incorporating semantic features that make use of external resources.

Unlike previous research on non-factoid CQA, the aim of answer summarization is to explore all useful information from candidate answers given a question. As far as we know, very little work on this task has been reported in the literature.

3. OVERVIEW OF THE METHOD

Before outlining our approach to the answer summarization problem, we first list the notations we use in this paper. See Table 1.

For each non-factoid CQA thread, we suppose there exists a question $q$ and a set of candidate answers $D = \{d_1, d_2, \ldots, d_D\}$, where each candidate answer $d \in D$ can be represented as a set of sentences and $D$ is the number of answers. We assume that, in total, there are $S$ sentences in the CQA thread, i.e., $S = \{s_1, s_2, \ldots, s_S\}$. Our task is to extract sentences from $S$ to construct the summary. To tackle the shortness, based on document expansion, we extend candidate answers by extracting $S'$ relevant sentences, represented as $S'$, from external knowledge material, e.g., Wikipedia. Totally, we have $\hat{S}$ sentences after document expansion, and we let $\hat{S} = S \cup S'$. Using deep neural network models, each sentence in both $S$ and $S'$, i.e., $\hat{S}$, can be represented as a $m$-dimensional vector. Therefore, after sentence vectorization, we have two vectors sets, $X$ and $X'$, corresponding to $S$ and $S'$, respectively.

A sparse coding-based method is proposed to reconstruct the semantic space of an aspect, revealed by answer sentences $\hat{S}$. A saliency score $a_s \in [0, 1]$ is determined for a sentence $s$ so as to define its contribution in constructing the semantic space of the topic from the answer content. For all candidate sentences $S$, we determine a saliency vector $A = [a_1, a_2, \ldots, a_S]$, in which each item corresponds to a candidate sentence. Given a question $q$, a sentence set $\hat{S}$, and a target summary length $L$, the goal of answer summarization in CQA is to select a subset of sentences $R \subseteq S$ such that the total number of words in $R$ is no more than $L$ while maximizing the sum of their saliency scores, i.e., $\sum_{s \in R} a_s$.

We provide a schematic overview of our method for performing answer summarization in Fig. 2. There are three main phases: (A) document expansion; (B) sentence representation; and (C) answer summarization. For all answers in $D$, in phase (A) (see §4.1), we obtain a set of candidate sentences $S$ with a set of auxiliary sentences $S'$. In (B) (see §4.2), given the set of sentences $S \cup S'$, we employ convolutional neural networks to obtain feature vectors $X$ and $X'$ to represent candidate sentences and auxiliary sentences, respectively. Then, based on these sentence vectors, in (C) (see §4.3) we generate the final answer summary $R$. We develop a sparse coding framework for this process: the first part in §4.3 optimizes the saliency vector $A$, where each element in $A$ is a saliency score.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>candidate answers</td>
</tr>
<tr>
<td>$V$</td>
<td>vocabulary in answers $D$</td>
</tr>
<tr>
<td>$E$</td>
<td>a knowledge base</td>
</tr>
<tr>
<td>$S$</td>
<td>candidate sentences</td>
</tr>
<tr>
<td>$S'$</td>
<td>auxiliary Wikipedia sentences</td>
</tr>
<tr>
<td>$X$</td>
<td>candidate sentence vectors</td>
</tr>
<tr>
<td>$X'$</td>
<td>auxiliary Wikipedia sentence vectors</td>
</tr>
<tr>
<td>$R$</td>
<td>a summary of answers, $R \subseteq S$</td>
</tr>
<tr>
<td>$A$</td>
<td>a saliency vector for candidate sentences</td>
</tr>
<tr>
<td>$D$</td>
<td>number of answers</td>
</tr>
<tr>
<td>$S$</td>
<td>number of sentences in answers</td>
</tr>
<tr>
<td>$S'$</td>
<td>number of auxiliary sentences from Wikipedia</td>
</tr>
<tr>
<td>$\hat{S}$</td>
<td>number of all sentences after document expansion</td>
</tr>
<tr>
<td>$L$</td>
<td>length limit of a summary of answers</td>
</tr>
<tr>
<td>$s$</td>
<td>a sentence</td>
</tr>
<tr>
<td>$q$</td>
<td>a question</td>
</tr>
<tr>
<td>$v$</td>
<td>an entity</td>
</tr>
<tr>
<td>$v_e$</td>
<td>an entity</td>
</tr>
<tr>
<td>$v_e$</td>
<td>an entity</td>
</tr>
<tr>
<td>$x_s$</td>
<td>a vector corresponding to a sentence $s$</td>
</tr>
<tr>
<td>$s$</td>
<td>a sentence</td>
</tr>
<tr>
<td>$a_s$</td>
<td>saliency score for a sentence $s$, $a_s \in A$</td>
</tr>
<tr>
<td>$m$</td>
<td>dimensionality of a sentence vector</td>
</tr>
<tr>
<td>$\lambda_s$</td>
<td>parameter for a sentence $s$ in sparse-coding</td>
</tr>
</tbody>
</table>

Table 1: Glossary.
for a candidate sentence; the second part in §4.3 generates the final answer summary $R$ using maximal marginal relevance (MMR) [2].

4. METHOD

In this section, we detail the steps in our approach to answer summarization.

4.1 (A) Document expansion

4.1.1 Entity linking

Given a candidate answer $d$, the target of entity linking is to identify the entity $e$ from a knowledge base $E$ that is the most likely referent of each sentence in $d$. A link candidate $e \in E$ links an anchor in $d$ to a target $g$, where an anchor is an n-gram token in an answer and each target $g$ is a Wikipedia article [29]. A target is identified by its unique title in Wikipedia. In the first step of entity linking, we identify as many link candidates as possible. We perform lexical matching of each n-gram anchor of answer $d$ with the target texts found in Wikipedia, resulting in a set of link candidates $E_d$ for each answer $d$. As the second step, after removing stop words, we combine words in each sentence $e \in d$, indicated as $V_e$, and the given question $q$, indicated as $V_q$, as a query $Q_e = \{V_e, V_q\}$ together to search relevant Wikipedia articles. We employ BM25 [36] to rank link candidates in $E_d$.

4.1.2 QA-based sentence ranking

Given a list of link candidates, we extract the most central sentences from the top three ranked Wikipedia articles. As in LexRank [8], Markov random walks are employed to iteratively optimize the ranked list of sentences, where each sentence may receive votes from other sentences. Suppose we collect a set of sentences, $S$, after entity linking. Firstly, we build the similarity matrix $M$, where each item in $M$ indicates the similarity between two sentences given $Q_e = \{V_e, V_q\}$ as a query. Given two sentences $s_i$ and $s_j$ in $S$, we compute the similarity $M_{i,j}$ between them as:

$$ M_{i,j} = \text{sim}(s_i, s_j | Q_e)/\sum_{s_k \in S} \text{sim}(s_i, s_k | Q_e). $$

At the beginning of the iterative process, an initial score for each sentence is set to 1/|S|, and at the $l$-th iteration, the score of $s'_l$ is calculated as follows:

$$ \text{score}(s'_l) = (1 - \lambda_e) \sum_{i \neq j} M_{i,j} \cdot \text{score}(s_j)' + \lambda_e \frac{1}{|S|}. $$

where $|S|$ equals the number of sentences in Wikipedia documents that have been linked to the anchor text $g$ (see §4.1.1) and the damping factor $\lambda_e = 0.15$. Then the transition matrix $M$ equals to:

$$ \tilde{M} = (1 - \lambda_e) M + \lambda e^T 1/|S|, $$

where $\tau$ is a column vector with all items equal to 1. The iterative process will stop when it converges. Since $\tilde{M}$ is a column stochastic matrix, the convergence of $\text{score}$ can be guaranteed, and a value of $\text{score}$ can be derived from the principle eigenvector of $\tilde{M}$ [8]. We extract the top sentences, denoted as $E_s$, from the resulting ranked list, and extend sentence $s$ with those $|E_s|$ sentences.

At the end of this document expansion phase, for the given set of candidate sentences $S$ in a CQA thread totally we extract an auxiliary set of sentences $S'$ from relevant Wikipedia articles.

4.2 (B) Sentence representation

Before sparse coding we transform the candidate sentences $S$ and auxiliary sentences $S'$ to basis vectors. To tackle the sparsity in candidate answer sentences, inspired by [39], we apply the convolutional neural networks (CNNs) to map sentences to distributed representations, by jointly considering the syntactic and semantic information hidden in sentences and the given question.

4.2.1 Convolutional neural networks

In recent years, convolutional neural networks [14, 16] have been proved to be effective in many language processing tasks, such as text classification [14] and short text ranking [39], by efficiently embedding sentences into low-dimensional vector space.

Given a sentence $s \in S \cup S'$, for the $i$-th word $v_i$ in $s$ we set $z_i \in \mathbb{R}^k$ to be a $k$-dimensional word vector. Suppose there are $n$ words in sentence $s$, then sentence $s$ can be represented as a vector $z_{1:n}$:

$$ z_{1:n} = z_1 \oplus z_2 \oplus \ldots \oplus z_n, $$

where $\oplus$ indicates the concatenation of two vectors. Let $z_{1+j-1}$ indicate the concatenation of word vectors $z_i, z_{i+1}, \ldots, z_{i+j-1}$; for these $j$ words within a window, we find a convolutional filter $w \in \mathbb{R}^{k \times j}$ to produce a new feature, i.e.,

$$ c_i = f(w^T \cdot z_{i:i+j-1} + b). $$

Here, $b$ is a bias parameter, and $f$ is set to be a non-linear function, such as the hyperbolic tangent or rectified linear function, to learn non-linear decision boundaries. Using this method, we can produce a feature map $e = [e_1, e_2, \ldots, e_{i+j}]$. Then a max-over pooling operation is applied over the feature map $e$ and the maximal element is extracted as the most important feature value, to be the feature corresponding to filter $w$. This pooling scheme naturally deals with variable sentence lengths. In CNNs, we can utilize multiple convolutional filters to get multiple features. Given $m$ filters and the sentence $s$, CNNs are able to build a rich feature representation $x_s$, which forms the penultimate layer and is passed to a fully connected softmax layer. The softmax layer computes the probability distribution over the labels, i.e.,

$$ p(y | x_s) = \frac{e^T \theta_y}{\sum_{y' \in Y} e^T \theta_{y'}}, $$

where $\theta_y$ is a weight vector of the label $y$ with $y = 1$ indicating that sentence $s$ is included in the answer summary and $y = 0$ indicating not included, and $Y = \{0, 1\}$ is the label set.

4.2.2 Feature vector generation

We consider each word in the given set of sentences as a $k$-dimensional vector, and initialize it by running word2vec tool [26] via the Yahoo Answers corpus.1 For all sentences in $S \cup S'$ we first generate the corresponding sentence matrices respectively. As we have discussed in §4.2.1, we utilize CNNs with $m$ filters in total to map a sentence $s \in S \cup S'$ to a $m$-dimensional vector $x_s$, and map the given question $q$ to a vector $x_q$. Following [39], we calculate the similarity between these two vectors as follows:

$$ \text{sim}(x_s, x_q) = x_q^T W x_s, $$

where $W$ indicates a similarity weight parameter in CNNs that is optimized during model training. Combining the similarity between a sentence $s$ and the given question $q$, we generate a joint layer

1https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
that concatenates intermediate vectors into a new vector, i.e., \( x_j = [x_s, \text{sim}(x_s, x_q), x_0] \). Based on this joint layer, the vector \( x_j \) is then passed through a fully connected hidden layer, which allows for modeling interactions between the components of the joined representation vector. We apply a non-linear function to transfer \( x_j \) into a new formulation, i.e.,

\[
y_j = f(w^T \cdot x_j + b),
\]

where \( w \) indicates an \( m \)-dimensional weight vector for the joint sentence vector \( x_j \). Thereafter, \( y_j \) moves to the softmax layer, as described in §4.2.1, to generate a probabilistic distribution over class labels. Here, we set \( y_s = 1 \) if sentence \( s \) is included in the answer summary. We assume that auxiliary sentences for \( s \) have the same class label as \( s \). We train the CNN model by optimizing an entropy-based loss function \( G \), as follows:

\[
G(Y, S \cup S', q|\Theta) = -\log \prod_{s \in S \cup S'} p(y_s|x_s, q) + \lambda_c \|\Theta\|_2^2,
\]

where \( \Theta \) indicates all parameters that are optimized by the CNN and \( \lambda_c \) is a free parameter. We optimize the parameters of the CNN using stochastic gradient descent (SGD), based on the back-propagation algorithm.

### 4.3 (C) Answer summarization

We propose an unsupervised sparse coding-based framework to tackle the answer summarization problem in CQA. An overview of our framework is depicted in Fig. 3, in which boxes indicate the question or answer sentences. The grey boxes indicate sentences that are selected as the summary.

The aim of sparse coding is to find a linear combination of basis vectors to minimize the reconstruction error function [18]. Using the trained CNN model from §4.2, each sentence \( s \in S \cup S' \) is mapped to an \( m \)-dimensional dense vector \( x_s \). Given the candidate vectors \( \mathcal{X} \) and auxiliary vectors \( \mathcal{X}' \), the target of our sparse coding answer summarization strategy is to find a set of candidate sentence vectors \( \mathcal{X}_R \subset \mathcal{X} \) that can be used to reconstruct the candidate sentence vectors \( \mathcal{X} \) and auxiliary sentence vectors \( \mathcal{X}' \), i.e., the sentences included in the summary.

Following the idea of sparse coding, we regard each candidate sentence vector \( x_i \in \mathcal{X} \) as a candidate basis vector, and all \( x_i \)'s are employed to reconstruct the semantic space of the aspect, including \( \mathcal{X} \) and \( \mathcal{X}' \). We propose the following preliminary error formulation:

\[
\frac{1}{2S} \sum_{i=1}^{S} \|x_i - \sum_{j=1}^{|R|} a_j x_j \|_2^2 +
\]

where the coefficient \( a_j \) is the saliency score for sentence vector \( x_j \). To harness the characteristics of the summarization problem setting more effectively, we refine the preliminary error formulation. To utilize the information contained in the question, we compute the cosine similarity \( \text{sim} \) between vectors representing each candidate sentence \( x_i \) and the question vector \( x_q \). Similarly, we introduce another parameter \( \text{sim}' \) to denote the cosine similarity between auxiliary sentence vector \( x'_s \) and \( x_q \). Because the summary sentences are sparse, we impose a sparsity constraint, \( \lambda_s > 0 \), on the saliency score vector \( A \) using the L1-norm, as a scaling constant to determine its relative importance.

Putting things together, we arrive at the following loss function:

\[
J = \min \frac{1}{2S} \sum_{i=1}^{S} \|x_i - \sum_{j=1}^{|R|} a_j x_j \|_2^2 + \frac{1}{2S'} \sum_{i=1}^{S'} \|x'_i - \sum_{j=1}^{\left|R\right|} a_j x_j \|_2^2 + \lambda_s \|A\|_1,
\]

subject to:
5. EXPERIMENTAL SETUP

In this section, we first formulate our research questions in §5.1 to guide our experiments. Then we describe our dataset in §5.2. The baselines and evaluation metrics are detailed in §5.3. The parameters for the proposed framework are listed in §5.4.

5.1 Research questions

To evaluate the effectiveness of our answer summarization method, we address the following research questions:

RQ1 How effective is our proposed method for answer summarization? Does it outperform state-of-the-art baselines?

RQ2 What is the impact of different representations on the overall answer summarization task?

RQ3 What is the impact of document expansion on the overall answer summarization task?

RQ4 What is the effect of the length constraint for the answer summary?

5.2 Dataset

We use a benchmark dataset released by Tomasoni and Huang [43]. Based on a Yahoo! Answers data collection with 89,814 question-answering threads, Tomasoni and Huang [43] removed factoid questions by applying a series of patterns for the identification of complex questions, and only leave non-factoid question-answering threads in the following patterns:

- Why, What is the reason […]
- How to, do, does, did […]
- How is, are, were, was, will […]
- How could, can, would, should […]

The ground truth of all these QA summaries is manually generated by selecting sentences subject to a fixed length limit of 250 words by human experts. In total, the dataset used in our experiments includes 100 non-factoid questions, 361 answers, 2,793 answer sentences, 59,321 words and 275 manually generated summaries.

5.3 Baselines and evaluation metrics

We write SPQAS for our sparse-coding based method as described in §4. To assess the effectiveness of our document expansion strategy, we write SPQAS-AT for our sparse-coding method restricted to the original answer texts only. We compare sentence representations based on our CNN-based architecture (in phase B) against word2vec as an alternative representation (SPQAS + word2vec). We perform comparisons between our methods and the following state-of-the-art baselines:

- MaQAS. [43], a metadata-aware question-answering summarization method;
- LexRank [8], a widely-used multi-document summarization model;
- SVM, A traditional supervised learning method that treats
question-answering summarization as a binary classification task;
• CNN [39], a state-of-the-art neural network architecture for question answer re-ranking;
• BestAns, a baseline that uses the top-ranked answer of the QA thread;
• Random, which extracts sentences randomly.

Following [43], we apply Porter stemming and set the length limit of the CQA answer summary to 250 words.

We adopt the ROUGE evaluation metrics [21], a widely-used recall-oriented group of metrics for document summarization that evaluates the overlap between a gold standard and candidate selections. We use ROUGE-1 (unigram based method), ROUGE-2 (bigram based method) and ROUGE-L (longest common subsequence) in our experiments. Statistical significance of observed differences between the performance of two approaches is tested using a two-tailed paired t-test and is denoted using ▲ (or ▼) for strong significance for α = 0.01; or ▼ (or △) for weak significance for α = 0.05.

5.4 Parameter settings

For the document expansion process (phase A), we set λc = 0.15, and extract 3 Wikipedia articles for each candidate sentence. For the vector generation process (phase B), we employ rectified linear units as the non-linear activation function, and the number of convolutional filter maps m is set to be 100. For the sparse coding model (phase C), we set the stop criteria T = 500, γ = 0.0001, and the learning rate η = 1 in Algorithm 1. For the sparsity item penalty, we set λs = 0.005.

6. EXPERIMENTAL RESULTS

In this section, we first consider the overall performance of our answer summarization method, SPQAS. Then we examine the impact of document expansion and of different sentence representations. Last we report on the impact of the length limit on answer summaries.

6.1 Overall performance

Firstly, we provide answer for RQ1. The evaluation results for non-factoid answer summarization using different models are summarized in Table 2. We find that SPQAS outperforms the other

<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>Random</td>
<td>0.534</td>
<td>0.418</td>
<td>0.525</td>
</tr>
<tr>
<td>LexRank</td>
<td>0.584</td>
<td>0.438</td>
<td>0.565</td>
</tr>
<tr>
<td>MaQAS</td>
<td>0.674</td>
<td>0.588</td>
<td>0.663</td>
</tr>
<tr>
<td>SVM</td>
<td>0.699</td>
<td>0.633</td>
<td>0.692</td>
</tr>
<tr>
<td>CNN</td>
<td>0.700</td>
<td>0.614</td>
<td>0.690</td>
</tr>
<tr>
<td>SPQAS</td>
<td>0.766▲</td>
<td>0.646▲</td>
<td>0.753▲</td>
</tr>
</tbody>
</table>

Table 2: Overall performance of all methods in answer summarization on ROUGE-1, ROUGE-2, and ROUGE-L. Statistically significant differences between SPQAS and the best performing baseline are marked in the upper right hand corner of SPQAS’s scores.

baselines, and significantly outperforms MaQAS in terms of all three ROUGE metrics. BestAns achieves the worst score since it suffers from a lack of diversity, which explains why we need to summarize all the answers for a given question. Since generic summarization methods neglect the correlation between a question and answers, the LexRank method does not perform well in the CQA answer summarization task. SPQAS also defeats SVM and CNN, two state-of-art supervised methods. While the difference between supervised methods (SVM and CNN) and unsupervised methods (LexRank and MaQAS) is always significant.

We take a closer look at SPQAS vs. the three top performing baselines. Against MaQaS, SPQAS offers relative performance improvements of 13.6%, 9.8%, and 13.6%, respectively, for the ROUGE-1, ROUGE-2 and ROUGE-L metrics. SPQAS outperforms SVM by 9.6%, 2.1%, and 8.8%, respectively, for ROUGE-1, ROUGE-2 and ROUGE-L. And the relative gains for SPQAS over CNN are 9.4%, 5.2%, and 9.1%. The two supervised methods, SVM and CNN, both perform better than the unsupervised MaQaS method, and the difference with MaQaS is statistically significant. Threads on which SPQAS is outperformed by one of the top 3 performing baselines tend to have longer questions.

6.2 Document expansion

To answer RQ2, i.e., to determine the impact of document expansion, we consider a variation of SPQAS that is restricted to the original answer texts only (essentially skipping phase A). See Table 3 for the results. As can be seen, the expanded documents lead to a 3% increase for all three ROUGE metrics, and SPQAS statistically significantly outperforms SPQAS-AT, which confirms the merit of utilizing document expansion in SPQAS. To gain a better understanding of the differences, we visualize the results in Fig. 4. Each of the red dots and the blue crosses denotes one question answer thread in our dataset. A QA thread is represented as a red dot when SPQAS outperforms SPQAS-AT on the thread, otherwise a blue cross. Clearly, the red dots outnumber the blue crosses for each metric, confirming that document expansion is helpful, although it is not a silver bullet: threads that have multiple and implicit aspects in answers are typically favored by SPQAS-AT, whereas many threads that have only few explicit aspect in answers cannot perform well after document expansion.

We further analyze the correlation between the length of answer texts and ROUGE scores; see Fig. 5. The average performance of the two methods appears to be similar when the total length of an answer thread is less than 800 words. SPQAS, i.e., with expanded texts, performs much better than SPQAS-AT with original answer texts only when the total length of the answer thread exceeds 800 words. As an aside, as the length of an answer thread increases, the performance of both methods, whether using expansions or not, decreases monotonously in terms of all metrics.

6.3 Sentence representations

To answer RQ3, we compare sentence representations based on our CNN-based architecture (in phase B) and word2vec as an alternative representation. Recall that SPQAS uses CNN. As an alternative, we consider SPQAS+word2vec, which produces sentence vectors using the idf-weighted sum of word vectors proposed in [26]
for the questions, candidate answer sentences, and auxiliary Wikipedia sentences. These sentence vectors are subsequently fed to the rest of the SPQAS pipeline (phase C) to generate answer summaries. The ROUGE scores for the methods based on these two kinds of sentence vectors are listed in Table 4. The results based on

Table 4: ROUGE results of different sentence representations. SPQAS+word2vec denotes SPQAS in which CNN-based sentence representations have been replaced by word2vec-based sentence representations. Statistically significant differences between SPQAS and SPQAS+word2vec are marked in the upper right hand corner of SPQAS’s scores.

<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPQAS+word2vec</td>
<td>0.729</td>
<td>0.612</td>
<td>0.717</td>
</tr>
<tr>
<td>SPQAS</td>
<td>0.766*</td>
<td>0.646*</td>
<td>0.753*</td>
</tr>
</tbody>
</table>

idf-weighted word2vec sentence vectors are satisfactory, beating the baselines listed in Table 2, but the CNN-based sentence vectors always perform statistically significantly better. This illustrates that the CNN-based sentence modeling architecture has a positive impact on sentence representation. And it also suggests that the SPQAS pipeline offers good robustness and scalability.

6.4 Length analysis

To answer RQ4, we vary the maximum allowed length of the answer summary and observe the performance of SPQAS; see Fig. 6, where we plot ROUGE-1 scores (recall, precision and F) for different length upper bounds. Generally, as we increase the upper bound on the length, recall rises gradually and precision decreases simultaneously, which is not surprising. A good balance between recall and precision can be found when the length constraint is set to between 200 and 300 words: this is where F-measure plateaus, with gains in recall balanced by losses in precision. Since short summaries (of equal quality) are preferred over longer ones, the
main lesson here is that our default length of 250 words is a sensible choice, offering both enough information and good readability.

7. CONCLUSION AND FUTURE WORK

We consider the task of answer summarization for non-factoid community question-answering. We identify the main challenges: the shortness and sparsity of answers, and the diverse aspect distribution. We utilize a document expansion strategy to enrich short texts using Wikipedia articles. After generating the distributed sentence representations via convolutional neural networks, we pursue a sparse coding-based framework to predict the saliency score of each candidate sentence, by jointly considering the candidate sentences and auxiliary Wikipedia sentences as reconstruction items in a novel loss function. We utilize a coordinate descent method to optimize our target loss function. We have demonstrated the effectiveness of our proposed method by showing significant improvements over multiple baselines tested with a benchmark dataset.

Limitations of our work include its ignorance of syntactic information and of semantic dependencies among answers. We also find that our method does not perform so well on answers with only few explicit aspect. Both are obvious areas for future work.

As to other areas of future work, given the positive impact of our sentence modeling techniques (in phase (B) of SPQAS), we believe there is more room to capture and exploit semantic information to further improve our summarization results. Secondly, transferring our method to the cross-language answer summarization and online answer summarization setting should be given new insights. Thirdly, it is interesting to consider a personalized summarization task on question-answering communities, based on dynamic clustering approaches [20, 55]. Fourthly, generating concise and abstractive answer summaries is worth considering. Finally, supervised and semi-supervised learning can be considered for improving the performance of answer summarization.

Acknowledgements. This work is supported by the Natural Science Foundation of China (61672322, 61672324, 61272240), the Natural Science Foundation of Shandong province (2016ZRB01148, ZR2012FM01), the Fundamental Research Funds of Shandong University, Ahold Delhaize, Amsterdam Data Science, Blendle, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELLIAS, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.015, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.166, HOR-11-10, CI-14-25, 652.002.001, 612-001.551, 652.001.003, 314-98-071, the Yahoo Faculty Research and Engagement Program, Yandex, and the EPSRC grant EP/K031953/1 (“Early-Warning Sensing Systems for Infectious Diseases”). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

8. REFERENCES


