World Wide Web (2019) 22:499–515
https://doi.org/10.1007/s11280-018-0597-7

MARES: multitask learning algorithm for Web-scale real-time event summarization

Min Yang[1] · Wenting Tu[2] · Qiang Qu[1] · Kai Lei[3] · Xiaojun Chen[4] · Jia Zhu[5] · Ying Shen[6]

Received: 5 October 2017 / Revised: 6 February 2018 / Accepted: 24 May 2018 /
Published online: 2 June 2018

Abstract Automatic real-time summarization of massive document streams on the Web has become an important tool for quickly transforming theoverwhelming documents into a novel, comprehensive and concise overview of an event for users. Significant progresses have been made in static text summarization. However, most previous work does not consider the temporal features of the document streams which are valuable in real-time event summarization. In this paper, we propose a novel **M**ultitask learning **A**lgorithm for Web-

This article belongs to the Topical Collection: *Special Issue on Deep vs. Shallow: Learning for Emerging Web-scale Data Computing and Applications*
Guest Editors: Jingkuan Song, Shuqiang Jiang, Elisa Ricci, and Zi Huang

✉ Jia Zhu
jzhu@m.scnu.edu.cn

Min Yang
min.yang1129@gmail.com

Wenting Tu
tu.wenting@mail.shufe.edu.cn

Qiang Qu
qiang.qu@siat.ac.cn

Kai Lei
leik@pkusz.edu.cn

Xiaojun Chen
xjchen@szu.edu.cn

Ying Shen
shenying@pkusz.edu.cn

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

[2] Department of Computer Science, Shanghai University of Finance and Economics, Shanghai, China

[3] School of Electronics and Computer Engineering, Peking University, ShenZhen, China

scale **R**eal-time **E**vent **S**ummarization (*MARES*), which leverages the benefits of supervised deep neural networks as well as a reinforcement learning algorithm to strengthen the representation learning of documents. Specifically, *MARES* consists two key components: (i) A relevance prediction classifier, in which a hierarchical LSTM model is used to learn the representations of queries and documents; (ii) A document filtering model learns to maximize the long-term rewards with reinforcement learning algorithm, working on a shared document encoding layer with the relevance prediction component. To verify the effectiveness of the proposed model, extensive experiments are conducted on two real-life document stream datasets: TREC Real-Time Summarization Track data and TREC Temporal Summarization Track data. The experimental results demonstrate that our model can achieve significantly better results than the state-of-the-art baseline methods.

Keywords Multitask learning · Real-time event summarization · Relevance prediction · Document filtering

1 Introduction

Real-time event summarization of massive document streams is a significant research challenge for natural language processing. In the past decades, a flurry of studies have been conducted on automatic event summarization. In general, existing event summarization approaches can be categorized as extractive and abstractive. The extractive summarization selects representative sentences from the input, while the abstractive summarization generates new phrases and sentences that may not appear in the original documents. Extractive summarization often achieves better results compared to abstractive summarization. This is due to the problems in abstractive summarization, such as semantic representation, inference and natural language generation, are relatively harder than data-driven approaches such as sentence extraction and ranking. In fact, the abstractive summarization has not reached to a mature stage yet. In this paper, we focus specifically on extractive summarization.

Extractive summarization is the task to identify important summary sentences from one or multiple documents such that the chosen sentences adequately describe the content subject to some specific requirements and constraints. Many techniques have been developed for extractive summarization such as graph-based [24], integer linear programming (ILP) [31], submodular and machine learning approaches [20, 26]. However, these methods do not consider the temporal features of the document streams that are valuable in practice. In static summarization the summary is only generated once, without being capable of summarizing events over time or updating the summary when new information emerges. Thus these methods are inadequate for online event summarization.

Real-time summarization tries to solve the aforementioned issue by providing updates to previous summaries, which only contain new or changed information that the users might want to know while avoiding information redundancy and latency in reporting. When performing summarization over several subsequent time windows of documents

[4] School of Computing Science, Shenzhen University, Shenzhen, China

[5] School of Computing Science, South China Normal University, Guangzhou, China

[6] School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, China

streams, a temporal summary can be created and updated in regular time intervals. Real-time summarization systems are highly attractive since they can be widely used in real-life scenarios by querying the social media streaming APIs. Despite its usefulness, the real-time summarization has received little attention in previous studies.

Typically, the real-time summarization could be divided into two phases. The first phase is relevance prediction, considering query–document textual similarity and capturing relevance signals contained in the temporal sequences of the stream [33]. The second phase is real-time document filtering, which aims to provide real-time event-specific updates of streaming text data that are timely, relevant, novel and comprehensive while avoiding redundancy. However, most approaches consider the two tasks as a sequential pipeline by first evaluating the relevance scores and then filtering the documents to provide notification pushing [42]. Nevertheless, these two subtasks are often mutually dependent. Knowing the relevance score of the document-query pair can help to filter the documents and skip the ones that are irrelevant to the query. Meanwhile, if we know whether a sentence is in the summary or not, the performance of relevance prediction may further be improved since the sentences included in the summary are usually intensely relevant to the query.

In this paper, we propose a novel **M**ultitask learning **A**lgorithm for Web-scale **R**eal-time **E**vent **S**ummarization (*MARES*), which simultaneously optimizes two coupled objectives: relevance prediction and real-time document filtering, with the hope that by explicitly exploiting their coupled relation, one can safeguard the performance of document filtering, improving over methods which do not consider such relation. *MARES* is a multi-task system, in which a document modeling module is shared across tasks. In particular, a relevance prediction classifier —the first subtask in the proposed framework —is trained with a hierarchical LSTM model. It helps the semantics analysis and comprehension of documents. For the second subtask, we do not simply treat real-time document filtering as a binary classification problem (select or skip) since pushing real-time text streams is not a short-term process but a dynamic forward decision process. In fact, successfully updating one document changes the true label (keep or skip) for documents that contain the same information but occur later in the stream. To maximize the long-term rewards and consider both historical dependencies and future uncertainty, we integrate reinforcement learning into our deep neural network for document filtering. The proposed model thus

The contributions of our model can be summarized as follows.

- To best of our knowledge, we are the first to use multi-task learning framework to simultaneously optimize the relevance prediction and document filtering tasks, in which the document representations are shared across tasks. These two tasks are tightly related and their correlation can be explicitly exploited from the document modeling perspective.
- Our model leverages the benefits of both the supervised deep neural networks and the reinforcement learning algorithm to strengthen the representation learning of documents.
- Our model uses reinforcement learning to optimize for highly rewarded summaries that are timely, relevant, novel and comprehensive while avoiding redundancy.
- To verify the effectiveness of the proposed model, we conduct extensive experiments on two widely used real-life datasets. The experimental results manifest that our model has significant superiority over competitors and sets state-of-the-art.

The rest of the paper is organized as follows. In Section 2, we review the related work. Section 3 presents our multitask model in details. In Section 4, we describe the experimental

data, implementation details and evaluation metrics. We demonstrate and analyze the experimental results in Section 5. Section 6 concludes this paper and indicates the future work.

2 Related work

Due to the large volume of online text information, it is costly and nearly impossible to manually generate text summaries with the goal of staying up-to-date. As such, automatic text summarization systems have been an active research field. In this section, we mainly discuss (i) the related work to extractive summarization, and (ii) the work related to real-time summarization.

2.1 Extractive summarization

Because labeled training data is often scarce, unsupervised approaches have been widely used in extractive summarization. Earliest extractive summarization approaches extracted salient sentences from documents using simple statistical features such as term frequency (TF), term frequency inverse document frequency (TF-IDF) [19]. These features were used to determine which terms in the documents might be important and favourable to use in the summary. Clustering approaches were then used to group the related documents into the same cluster and produce the cluster-wise summary [8]. Greedy search played an important role in unsupervised extractive summarization, which was able to select a set of sentences that were of saliency and redundancy [16, 31]. Among them, McDonald [31] was the first to formulate the extractive summarization as an Integer Linear Programming (ILP) problem which assumed that the quality of a summary was proportional to the sum of the relevance scores of the chosen sentences, penalized by the sum of the redundancy scores of all pairs of selected sentences. However, these methods are not scalable above large volume of sentences. More recently, the graph theoretic approache [37] wes proposed to rank sentences and characterize various sentence relationships. Sentences in the documents are represented as nodes in an undirected graph. The nodes in the graph are connected with a common edge if the sentences share common words. The vertices with the high rank are important sentences, thus they are likely to be included in the summary. As an example, Erkan and Radev [10] proposed LexRank algorithm for computing sentence importance, where a connectivity matrix based on intra-sentence cosine similarity was used as the adjacency matrix of the graph representation of sentences.

In contrast to the aforementioned unsupervised summarization approaches, the machine learning approaches treated the extractive summarization process as a classification problem, where the classifiers are were trained to predict whether a sentence is in the summary or not. Research on machine learning has achieved great success on enhancing the models' accuracy and efficiency [46, 50–52]. Different classification algorithms have been explored, such as Bayesian method [25], Conditional Random Field [13], and Logistic Regression [26]. Inspired by the successes of deep neural networks in computer vision [14, 43, 49], natural language processing [7], and multimedia [15, 29, 38–40], several recent studies have been proposed to employ deep neural networks to summarization. Kageback et al. [20] exploited the effects of using phrase embeddings for summarization, and demonstrated that these can improve the performance of the previous methods. Cao et al. [4] developed a ranking framework based on recursive neural networks (RNN) to rank sentences for multi-document summarization. Cheng et al. [7] considered a single document summarization

as a sequence labeling task and modeled it with a hierarchical document encoder and an attention-based extractor. Cao et al. [5] leveraged a large amount of text classification data to boost the performance of multi-document summarization. The text classification and summarization tasks shared the same document representation learned via convolutional neural network (CNN).

2.2 Real-time summarization

The previous studies have concentrated mostly on offline scientific documents or newswire data. However, the events may constantly evolve over time [47]. There has been increasing interest in developing real-time text summarization systems to retain real-time properties of continuous document streams. Allen et al. [2] were the first one who explored the streaming or temporal summarization for topic detection and tracking. Guo et al. [17] proposed a temporal summarization method, which generated update summaries from news sources about critical events. Subsequently, temporal summarization has been operationalized in the TREC Temporal Summarization (TS) Tracks from 2013 to 2015 [1, 3, 45]. This task aimed to identify sentences that were relevant to specific breaking news stories which contained new and important content. The top performers at TREC TS are the ranking/MDS combination models [30], and the affinity propagation clustering model [21]. Furthermore, the real-time pushing on text streams has attracted increasing attention at the 2015 TREC Microblog Track [27] and the 2016 TREC Real-Time Summarization (RTS) Track [28]. For example, the winner of the real-time filtering task at TREC 2015 Microblog track [41] explored simple dynamic emission strategies to establish and maintain appropriate thresholds for pushing relevant tweets. Roegiest et al. [34] introduced an interleaved evaluation for prospective notification to examined user behavior in a realistic setting.

Recently, several studies regarded real-time summarization tasks as a dynamic forward decision process by using reinforcement learning techniques. The current decision would affect further decisions and the further document streams would create uncertainty on the current decision. Kedzie et al. [22] adopted a locally optimal learning to learn the policy for selecting or skipping a sentence in document streams. They casted streaming summarization as a form of greedy search and trained the model to imitate the behavior of an oracle summarization system. Tan et al. [42] used Q-Network that contained a Long Short Term Memory (LSTM) layer and three fully connected neural network layers to maximize the long-term rewards for the real-time summarization.

Different from the previous work, our model jointly predicts the document relevance and filters the documents for real-time notification pushing, instead of treating the two tasks as a sequential pipeline by first predicting the relevance score and then updating summary by filtering the documents.

3 Our model

We assume there are a set of document queries Q, where each query q can be a set of keywords or paragraph-length description, representing users' information needs.[1] The corpus consists of a document stream $D = \{d_1, d_2, \ldots, d_t, \ldots\}$, where each document is a post crawled from the Web such as the news article, forum data, Weblog post, or microblog post.

[1] An example of text queries can be found in the experimental part.

For simplicity, we assume a fixed length stream of size T but this is not strictly necessary. Our real-time summarization algorithm performs two tasks: 1) predict the relevance score y_t^r between the incoming document d and the query q at time step t; 2) select or skip (i.e., y_t^s) the current document d as the summary. The goal of our model is to produce a filtered stream of documents as the real-time summary which is relevant (to query q), timely, comprehensive, non-redundant, and novel.

In this section, we elaborate the two components of our real-time summarization method: the relevance prediction model and the document filtering model.

3.1 Relevance prediction

Relevance prediction is crucial for information retrieval from social media streams such as Twitter. Given the information need expressed as a query, we aim to assign a relevance label (i.e., highly-relevant, relevant or irrelevant) to each document of the stream, exploiting all historical documents in the stream. The relevance score can be beneficial for the final document filtering.

Inspired by the great success of LSTM in natural language processing, we employ a hierarchical LSTM to relevance prediction, which decomposes the document stream into a two-level hierarchy: a sequence of documents, where each document is a sequence of words.

Each LSTM unit at time t consists of a memory cell c_t, an input gate i_t, a forget gate f_t, and an output gate o_t. These gates are computed from the previous hidden states h_{t-1} and the current input x_t, which are computed by

$$i_t = \sigma(W_i x_i + U_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

where σ is the sigmoid function, $W_i, b_i, W_f, b_f, W_o, b_o$ are the parameters of input, forget and output gates.

The memory cell c_t is updated by partially forgetting the existing memory and adding a new memory content $\tilde{c}_t$:

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{4}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{5}$$

Once the memory content of the LSTM unit is updated, the hidden state of current timestamp is computed as:

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

These gates and the memory cell allow a LSTM unit to adaptively forget, memorize and expose the memory content. For simplicity, the updation of the hidden state of LSTM at time step t is denoted as

$$h_t = LSTM(h_{t-1}, x_t) \tag{7}$$

3.1.1 Word-level LSTM

The word-level LSTM converts each document d and query q into sequences of hidden states $h^d = \left\{h_1^d, \ldots, h_{N_d}^d\right\}$ and $h^q = \left\{h_1^q, \ldots, h_{N_q}^q\right\}$ respectively:

$$h_n^d = LSTM(h_{n-1}^d, w_n^d) \tag{8}$$

$$h_m^q = LSTM(h_{m-1}^q, w_m^q) \tag{9}$$

where w_n^d is the n-th word of the document d, w_m^q is the m-th word of the query q, h_n^d represents the n-th hidden state of the document d and acts as a vector representation of tokens seen up to position n in the document d, and h_m^q represents the m-th hidden state of the query q.

For the long sequence, the last state of the LSTM may not reflect important information seen at the beginning of the sequence. Thus, we adopt the bidirectional LSTM, which runs two chains: one forward through the input and another backward, i.e. reversing the tokens in the input sentence. In time step n, we summarize the information in the forward and backward LSTM hidden states of document d by taking the concatenation of the two LSTMs: $h_n^d = \left[\overrightarrow{h_n^d}, \overleftarrow{h_n^d}\right]$. Similarly, we can obtain the hidden state of the time step m of query q as $h_m^q = \left[\overrightarrow{h_m^q}, \overleftarrow{h_m^q}\right]$.

In particular, we may treat the last hidden state $h_{N_d}^d$ and $h_{N_q}^q$ as the vector representation of document d and query q, respectively. The document-query representation v associated with the document d and the query q at time step t are computed as:

$$v_t = g(V^d.h_{N_d}^d + V^q.h_{N_q}^q) \tag{10}$$

where V^d and V^q are projection parameters to be learned, g is a non-linear function (i.e., Rectified Linear unit, ReLu).

3.1.2 Document-level LSTM

The document-level LSTM models the past documents and queries by processing iteratively each document-query representation. The document-level LSTM is structurally similar to the word-level LSTM. Formally, the hidden state of the document-level LSTM at time step t is:

$$z_t = LSTM(z_{t-1}, v_t) \tag{11}$$

where v_t is the document-query representation at time step t, z_t represents the summary of the historical information up to time step t which is subsequently fed to the document-level LSTM for predicting the relevance. The document-level LSTM keeps track of the past documents in the stream by processing each document iteratively.

The outputs of the document-level LSTM are fed into a fully-connected layer and a softmax classifier, resulting the relevance label of document d corresponding to query q:

$$f_t^r = \varrho(V_1^r \cdot z_t) \tag{12}$$

$$\hat{y_d^r} = softmax(V_2^r \cdot f_t^r) \tag{13}$$

where V_1^r and V_2^r are projection parameters, ϱ is the activation function for the fully connected neural network (i.e., Rectified Linear unit, ReLu).

3.2 Real-time document filtering

Real-time document filtering of document streams plays a significant role in assisting users in acquiring timely and relevant information in social media. In this paper, we formulate the real-time pushing on text stream as real-time decision making task —push or skip the incoming document, considering the novelty, comprehensiveness, immediacy, and relevance to the given query while avoiding redundancy.

As Figure 1 depicts, the document filtering subtask shares the same document representation module with the relevance prediction subtask. In this paper, we formulate the real-time document filtering as a long-term optimization problem, which selects or skips each document d as it is observed and aims to provide users a filtered stream of documents that are comprehensive, low redundant, timely and relevant to the given query q.

We propose a policy gradient reinforcement learning algorithm to maximize the long-term rewards by considering both historical dependencies and future uncertainty of the document streams. Instead of learning an approximation of the underlying value function [42], our policy gradient algorithm attempts to search the policy space directly in order to follow the gradient of the long-term rewards. In addition to being able to express stochastic optimal policies and being robust to small changes in the underlying function approximation, policy gradient algorithm is guaranteed to converge under appropriate conditions. We first introduce the state, action, policy and reward of the reinforcement learning architecture below.

3.2.1 State

In time step t, a state s_t is denoted by the previous documents up to time step t. In order to take advantage of generalization with the deep neural networks, the historical documents are transformed to a state vector representation z_{t-1} by (11) and (10).

3.2.2 Action

The action at time step t is $a_t \in \{0, 1\}$ with 1 indicating we push the t-th document to our update summary, and 0 indicating we skip it.

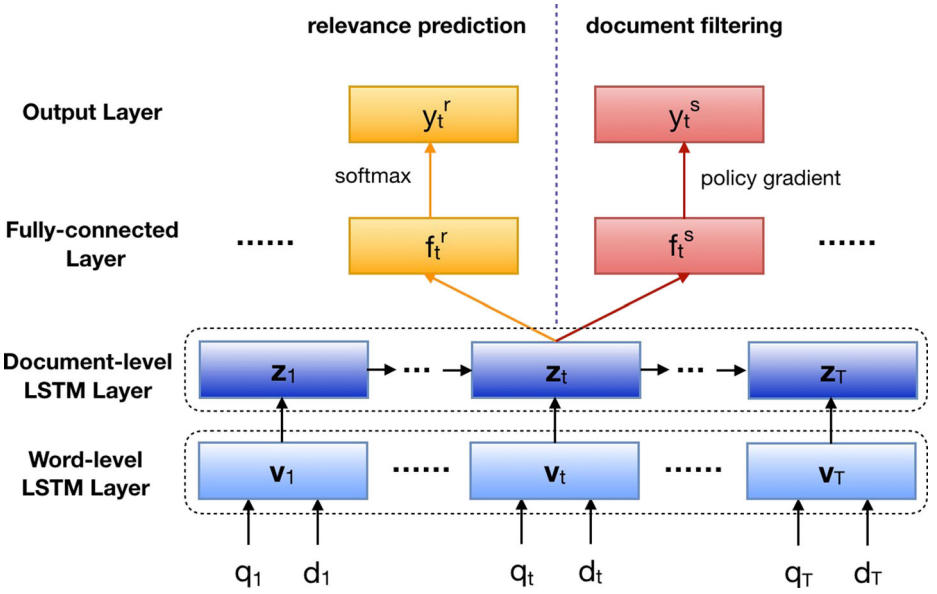

Figure 1 Architecture and dataflow chart of MARES

3.2.3 Policy

We employ stochastic policy gradient method to approximate a stochastic policy π_θ directly using an independent function approximator with its own parameters θ, whose input is state s_t and current document-query representation v_t, whose output is the probabilities to choose action a_t. Mathematically, the policy function can be defined as:

$$f_t^s = g(V_1^p \cdot z_t) \tag{14}$$

$$\pi_\theta(a_t|s_t) = softmax(V_2^p \cdot f_t^s) \tag{15}$$

where V_1^p and V_2^p are projection parameters, g is a non-linear function (i.e., Rectified Linear unit, ReLu).

3.2.4 Reward

The policy gradient algorithm is a type of reinforcement learning methods, which relies upon optimizing parametrized policy with respect to the expected return (long-term cumulative rewards) by gradient descent. When we reach the end of the sequence, we get a reward of $R(a_T)$**,** which represents the score for producing the action sequence $a_{1:T}$ given document stream and the update summary.

In quintessential reinforcement learning setting, an agent receives rewards at each intermediate step while future rewards are discounted to balance short-term and long-term gain. In our case, however, the agent receives zero reward during intermediate steps, observing a reward only at the end. To alleviate the absence of intermediate reward, we estimate the value of the partial sequence via Monte Carlo search with the roll-out policy, following the strategy as in [48]. Concretely, at each time step t, the unknown last $T - t$ actions are sampled via Monte Carlo search, and the whole sequence is then evaluated for intermediate reward. This strategy significantly improves the convergence speed, training stability and robustness of our model. In this paper, we use our policy gradient method to optimize two kinds of rewards: expected gain (*EG*, see (22)) and normalized cumulative gain (*nCG*, see (23)). The final reward for action a_t is a weighted combination of these two rewards:

$$R(a_t) = \lambda EG(a_{1:t-1}; a_t; a_{t+1:T}) + (1 - \lambda)nCG(a_{1:t-1}; a_t; a_{t+1:T}) \tag{16}$$

where $R(a_t)$ denotes the reward resulting from action a_t, λ is the hyper-parameter that determines the wights of EG and nCG.

The objective to maximize is the expected future reward:

$$J_{RL}(\theta) = \mathbb{E}_{\pi_\theta(a_t|s_t)}\left[\sum_{t=1}^{T} R(a_t)\right] \tag{17}$$

3.3 Joint training

Overall, our model consists of two subtasks, each has a training objective. For the relevance prediction subtask, given a labeled training set$(X, Y^r) = \{(x_1, y_1^r), \ldots, (x_T, y_T^r)\}$,

we minimize directly the cross-entropy between the predicted label $\hat{y^r}$ and the ground truth y^r. The loss objective is defined as:

$$L_1(\theta) = -\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{K}\mathbf{I}\left\{y_t^r = i\right\}\log(\hat{y_t^r}) \tag{18}$$

where $\mathbf{I}\{\cdot\}$ is an indicator such that $\mathbf{I}\{\text{true}\}=1$ and $\mathbf{I}\{\text{false}\}=0$.

For the real-time document filtering subtask, we optimize the expected future reward (see (17)), which is equivalent to minimizing the negative future reward

$$L_2(\theta) = -\mathbb{E}_{\pi_\theta(a_t|s_t)}\left[\sum_{t=1}^{T} R(a_t)\right] \tag{19}$$

For the purpose of learning better representations of query-document pairs, we optimize the two coupled objectives simultaneously. The joint multi-task objective function is minimized by:

$$L(\theta) = \lambda_1 L_1(\theta) + \lambda_2 L_2(\theta) \tag{20}$$

where λ_1 and λ_2 are hyper-parameters that determines the weights of L_1 and L_2.

3.3.1 Updating model parameters

For the relevance prediction model, we can compute the derivative of L_1 with respect to each parameter that is to be learned by using the chain rule directly. The reader can refer to [6] for details of deriving backpropagation for LSTM network.

According to the policy gradient theorem [44], we have the following derivations for L_2 with respect to parameters θ of the model:

$$\begin{aligned}\nabla_\theta L_2(\theta) &= -\nabla_\theta\left[\sum_{a_t\in A}\pi_\theta(a_t|s_t)\sum_{t=1}^{T}R(a_t)\right] \\ &= -\sum_{a_t\in A}\nabla_\theta\log\pi_\theta(a_t|s_t)\sum_{t=1}^{T}R(a_t)\end{aligned} \tag{21}$$

where A is the action space. The reader can refer to [44] for details of policy gradient theorem.

After getting the stochastic gradient of $L(\theta)$ with respect to the set of parameters θ, we use ADAM [23] optimization algorithm to update the parameters.

4 Experimental setup

4.1 Datasets

To evaluate the effectiveness of our model, we conduct extensive experiments on two publicly available datasets: TREC 2016 Real-Time Summarization Track data (TREC-RTS)[2] and TREC 2014 Temporal Summarization Track data (TREC-TS).[3]

[2] http://trecrts.github.io/

[3] http://www.trec-ts.org/

4.1.1 TREC-RTS

This dataset contains 61,519 tweets that are offered by Twitter streaming API (approximately 1% samples of all tweets) during the period from August 2, 2016 to August 11, 2016. Each tweet is associated with a relevance label towards its corresponding interest profile (topic): *highly relevant*, *relevant* and *non-relevant*. During the evaluation period, 56 interest profiles (topics) are monitored, representing users' information needs, where each interest profile is comprised of a *title*, *description* and *narrative* fields. The *title* contains a short description of the information need, and the *description* and *narrative* are sentence- and paragraph-long elaborations of the information need, respectively (see Table 1). Following [42], we randomly select tweets of 41 interest profiles as training data, tweets of 5 interest profiles as the validation data, and the remaining are used for testing.

4.1.2 TREC-TS

The original TREC-TS dataset consists of approximate 1.2 billion timestamped documents crawled from the Web between from October, 2011 and February 2013 [11]. This corpus is the largest dataset that is widely used for real-time summarization, which is composed of news articles, Weblogs, forum posts and other Web pages. We clean the data by only keeping the visible text of apparently English documents, belonging to 44 events (topics) where each event (topic) described by a set of keyword queries or short descriptions. The cleansed version contains over 400 millions documents. Similar to TREC-RTS dataset, annotators have generated relevance judgment (*highly relevant*, *relevant* and *non-relevant*) for each document based on their personal interpretation of the topic. Since the document stream of each training query is too large to be used directly in our combinatorial search space. In order to make training time reasonable yet representative, we downsample each stream to a length of 100 sentences, similar to [22]. The downsampling is done uniformly over the entire stream. We randomly select the documents of 34 topics as the training data, documents of 5 topics as the validation data, and the remaining are used for testing.

4.2 Baseline methods

In the experiments, we evaluate and compare our model with the state-of-the-art document filtering and real-time summarization approaches, which we describe below:

4.2.1 Document filtering baselines

LSTM LSTM is a widely-used baseline method to learn sentence representations and has been proven effective in many natural language processing tasks.

Table 1 A example of query randomly selected from TREC 2016 RTS dataset

Query	{ "topid" : "RTS8", "title" : "Donald Trump's tax policy", "description" : "Find information on U.S. presidential candidate Donald Trump's policies on how he plans to tax America.", "narrative" : "The user is an American concerned about the upcoming presidential election and wants to understand Donald Trump's position on taxes." }

Kernel density estimation (KDE) This method characterized the temporal density function with kernel density estimation [9].

Sentence model CNN (SM-CNN) The main idea of this method is to develop two distributional sentence models based on convolutional neural networks [36]. We use the publicly available code[4] to implement this model.

Multi-perspective CNN (MP-CNN) It uses a convolutional neural network (CNN) to extracts features at multiple levels of granularity and employs multiple types of pooling [18]. The method is run by using the publicly available implementation.[5]

4.2.2 Real-time summarization baselines

Interest profile similarity (IPS) This method updates the summary by choosing the documents whose relevance score are higher than a fixed threshold.

Affinity propagation (AP) This method employs affinity propagation clustering algorithm [12] to update the summary by choosing the documents that are cluster center.

Learning to search (LS) It achieves state-of-the-art result in real-time summarization [22], which adopts the *learning to search* approach [35] - a reinforcement learning algorithm for structured prediction problems.

Cosine similarity threshold (CST) It is one of the top performers in TREC 2015 temporal-summarization track [21], which selects the documents that are below a cosine similarity threshold to any of the chosen updates.

Neural network based reinforcement learning (NNRL) This method is proposed in [42], in which a Q-Network is used to learn an optimal policy which maximizes the long-term rewards.

4.3 Implementation details

In the experiments, we use 100-dimensional word2vec [32] vectors to initialize the word embeddings for words in the datasets. We initialize the recurrent weight matrices as random orthogonal matrices and all the bias vectors are initialized to zero. The hidden state size of LSTM is 500. We conduct mini-batch (with size 64) training using Adam [23] optimization algorithm to train the model. L_2 regularization (with a weight decay value of 0.001) and dropout (with a dropout rate of 0.2) are used on the weights and biases of the LSTM layer to avoid overfitting. We set λ=0.5, λ_1 =0.5, λ_2 = 0.5.

4.4 Evaluation metrics

For relevance prediction, we use prediction accuracy as the evaluation metric.

For document filtering, we adopt the official evaluation metrics of TREC 2016 Real-time Summarization Track, including Expected Gain (EG), Normalized Cumulative Gain (nCG),

[4] https://github.com/castorini/SM-CNN-Torch

[5] https://github.com/castorini/MP-CNN-Torch

Gain Minus Pain (GMP), and latency. All these metrics are computed for each interest profile per day and averaged.

4.4.1 Expected gain (EG)

The EG score for an interest profile on a particular day is defined as:

$$EG = \frac{1}{N} \sum G(t) \tag{22}$$

where N is the number of tweets returned, and $G(t)$ is the gain of each tweet at time t. Three levels of gains (G) are assigned to a tweet with scores from 0 to 1, where 1 denotes *highly-relevant*, 0.5 denotes *relevant*, and 0 denotes *not relevant*. Once a tweet from a cluster is retrieved, all other tweets from the same cluster automatically become not relevant. This penalizes systems for returning redundant information.

4.4.2 Normalized cumulative gain (nCG)

The nCG score is defined as:

$$nCG = \frac{1}{Z} \sum G(t) \tag{23}$$

where Z is the maximum possible gain.

In particular, for *silent day* (the day in which there is no relevant tweets), the EG-1 and nCG-1 variants have a score of 1 if the model does not push any documents, or 0 otherwise. In the EG-0 and nCG-0 variants, for *silent day*, the gain is 0.

4.4.3 Gain minus pain (GMP)

The GMP aims to penalize the system for updating non-relevant documents, which is defined as:

$$GMP = \lambda G - (1 - \lambda) P$$

where G (gain) is defined in the same manner as above, P is the number of non-relevant tweets that are pushed, and λ controls the balance between the two. In this paper, we set $\lambda = 0.5$.

4.4.4 Latency

Instead of applying a latency penalty to gain, the latency is calculated separately. The latency score only computed on the tweets that are relevant. The mean difference between the time the tweet was pushed and the first tweet in the semantic cluster that the tweet belongs are reported.

5 Experimental results

In this section, we report the system comparisons for both the relevance prediction subtask and the document filtering subtask.

Table 2 The accuracy scores of relevance prediction

Dataset	LSTM	TDE	SM-CNN	MP-CNN	MARES
TREC-RTS	0.688	0.758	0.744	0.786	0.792
TREC-TS	0.724	0.766	0.775	0.792	0.825

Table 3 Document filtering evaluation results on TREC-RTS dataset

Method	EG-0	nCG-0	EG-1	nCG-1	GMP	Latency
IPS	0.0334	0.0397	0.2006	0.2125	−0.3244	192344
AP	0.0372	0.0311	0.2321	0.2352	−0.1034	134077
CST	0.0483	0.0631	0.2622	0.2544	−0.3214	91456
LS	0.0703	0.0809	0.2711	0.2968	−0.1884	85665
NNRL	0.0691	0.0846	0.2816	0.2971	−0.2345	84343
MARES	0.0785	0.0934	0.2889	0.3085	−0.0855	73572

Table 4 Document filtering evaluation results on TREC-ST dataset

Method	EG-0	nCG-0	EG-1	nCG-1	GMP	Latency
IPS	0.0424	0.4729	0.2371	0.2342	−0.2565	126329
AP	0.0397	0.4481	0.2554	0.2122	−0.1434	135334
CST	0.0466	0.5015	0.2399	0.2504	−0.3313	98045
LS	0.0521	0.5217	0.2601	0.2601	−0.0979	78433
NNRL	0.0517	0.5008	0.2713	0.2544	−0.1091	74257
MARES	0.0544	0.5721	0.2847	0.2619	−0.0966	65946

Table 5 Ablation study for the TREC-RTS dataset

Method	EG-0	nCG-0	EG-1	nCG-1	GMP	Latency
MARES	0.0785	0.0934	0.2889	0.3085	−0.0855	73572
MARES w/o RP	0.0734	0.0883	0.2827	0.2948	−0.1366	80145
MARES w/o RL	0.0741	0.0894	0.2853	0.2979	−0.1195	79638

Table 6 Ablation study for the TREC-ST dataset

Method	EG-0	nCG-0	EG-1	nCG-1	GMP	Latency
MARES	0.0544	0.5721	0.2847	0.2619	−0.0966	65946
MARES w/o RP	0.0513	0.5336	0.2755	0.2584	−0.1024	73842
MARES w/o RL	0.0525	0.5429	0.2782	0.2591	−0.0998	71225

5.1 Relevance prediction

To test the performance of our model on relevance prediction, we first report the relevance prediction accuracy in Table 2. We observe that for TREC-RTS dataset, our model performs on par with or slightly better than the other models. While for TREC-TS dataset, our model achieves a noticeable improvement. This is within our expectation since TREC-TS contains a greater amount of documents, our multitask model is more powerful in learning the document representations.

5.2 Document filtering

In this experiment, we evaluate the performance of real-time document filtering. The experimental results are summarized in Tables 3 and 4, which show that our model substantially and consistently outperforms the baseline methods by a noticeable margin on all the evaluation metrics. Specifically, *MARES* exhibits higher expected gain than the competitors on both datasets by a significant margin, indicating the novelty and comprehensiveness of the updated summary. For relevance, *MARES* achieves highest GMP scores, which suggests that less fraction of non-relevant documents are selected by our model. In addition, the low latency score verifies the timeliness of our model. The main advantage of MARES comes from its capability of leveraging the benefit of supervised deep neural networks as well as a reinforcement learning algorithm to strengthen the representation learning of documents.

5.3 Ablation study

In order to analyze the contribution of each subtask to the superiority of *MARES*, we also report the ablation test in terms of discarding the relevance prediction task and the reinforcement learning algorithm, respectively. The results are demonstrated in Tables 5 and 6. Not surprisingly, both factors contribute great improvement to the *MARES*.

6 Conclusion and future work

We proposed a novel multi-task framework for real-time event summarization, which treated relevance prediction and document filtering as two subtasks, strengthening the representation learning in document modeling. A reinforcement learning algorithm was designed to maximize the long-term rewards for real-time summarization. Experiments on two real-life datasets showed the superiority of the proposed model. In the future, we would like to adapt the proposed model to real-time abstractive event summarization, which generates a short and concise summary that captures the salient ideas of the source input.

Acknowledgements This work was partially supported by the National Science Foundation of China (No.61750110516), the Shenzhen Key Fundamental Research Projects (Grant No. JCYJ20170412150946024), and the CAS Pioneer Hundred Talents Program.

References

1. Aliannejadi, M., Bahrainian, S.A., Giachanou, A., Crestani, F.: University of lugano at trec 2015: Contextual suggestion and temporal summarization tracks. In: TREC (2015)

2. Allan, J., Gupta, R., Khandelwal, V.: Temporal summaries of new topics. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 10–18. ACM (2001)
3. Aslam, J., Diaz, F., Ekstrand-Abueg, M., McCreadie, R., Pavlu, V., Sakai, T.: Trec 2014 temporal summarization track overview. Technical report (2015)
4. Cao, Z., Wei, F., Li, D., Li, S., Zhou, M.: Ranking with recursive neural networks and its application to multi-document summarization. In: AAAI, pp. 2153–2159 (2015)
5. Cao, Z., Li, W., Li, S., Wei, F.: Improving multi-document summarization via text classification. In: AAAI, pp. 3053–3059 (2017)
6. Chen, G.: A gentle tutorial of recurrent neural network with error backpropagation. arXiv:1610.02583 (2016)
7. Cheng, J., Lapata, M.: Neural summarization by extracting sentences and words. arXiv:1603.07252 (2016)
8. Deshpande, A.R., Lobo, L.M.R.J.: Text summarization using clustering technique. Int. J. Eng. Trends Technol., **4**(8) (2013)
9. Efron, M., Lin, J., He, J., De Vries, A.: Temporal feedback for tweet search with non-parametric density estimation. In: SIGIR, pp. 33–42. ACM (2014)
10. Erkan, G., Radev, D.R.: Lexrank: Graph-based lexical centrality as salience in text summarization. J. Artif. Intell. Res. **22**, 457–479 (2004)
11. Frank, J.R., Kleiman-Weiner, M., Roberts, D.A., Niu, F., Ce, Z., Ré, C., Soboroff, I.: Building an entity-centric stream filtering test collection for trec. Technical report. Massachusetts. Inst of. Tech. Cambridge. (2012)
12. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
13. Galley, M.: A skip-chain conditional random field for ranking meeting utterances by importance. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp, 364–372. Association for Computational Linguistics (2006)
14. Gao, L., Guo, Z., Zhang, H., Xing, X., Shen, H.T.: Video captioning with attention-based lstm and semantic consistency. IEEE Trans. Multimed. **19**(9), 2045–2055 (2017)
15. Gao, L., Song, J., Liu, X., Shao, J., Liu, J., Shao, J.: Learning in high-dimensional multimedia data: the state of the art. Multimed. Syst. **23**(3), 303–313 (2017)
16. Gillick, D., Favre, B.: A scalable global model for summarization. In: Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing, pp. 10–18. Association for Computational Linguistics (2009)
17. Guo, Q., Diaz, F., Yom-Tov, E.: Updating users about time critical events. In: European Conference on Information Retrieval, pp. 483–494. Springer (2013)
18. He, H., Gimpel, K., Lin, J.J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: EMNLP, pp. 1576–1586 (2015)
19. Hovy, E., Lin, C.-Y.: Automated text summarization and the summarist system. In: Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998, pp. 197?214. Association for Computational Linguistics (1998)
20. Kågebäck, M., Mogren, O., Tahmasebi, N., Dubhashi, D.: Extractive summarization using continuous vector space models. In: Proceedings of the 2nd EACL Workshop on Continuous Vector Space Models and their Compositionality, pp. 31–39 (2014)
21. Kedzie, C., McKeown, K., Diaz, F.: Predicting salient updates for disaster summarization. In: ACL (1), pp. 1608–1617 (2015)
22. Kedzie, C., Diaz, F., McKeown, K.: Real-time Web scale event summarization using sequential decision making. In: International Joint Conference on Artificial Intelligence, pp. 3754–3760 (2016)
23. Kingma, D., Adam, J.B.: A method for stochastic optimization. arXiv:1412.6980 (2014)
24. Kruengkrai, C., Jaruskulchai, C.: Generic text summarization using local and global properties of sentences. In: IEEE/WIC International Conference on Web Intelligence, 2003. Proceedings, pp. 201–206. IEEE (2003)
25. Kupiec, J., Pedersen, J., Chen, F.: A trainable document summarizer. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 68–73. ACM (1995)
26. Li, C., Qian, X., Liu, Y.: Using supervised bigram-based ilp for extractive summarization. In: ACL, pp. 1004–1013 (2013)
27. Lin, J., Efron, M., Wang, Y., Sherman, G.: Overview of the trec-2015 microblog track. Technical report (2015)

28. Lin, J., Roegiest, A., Tan, L., McCreadie, R., Voorhees, E., Diaz, F.: Overview of the trec 2016 real-time summarization track. In: TREC (2016)
29. Liu, G., Yan, Y., Subramanian, R., Song, J., Guoyu, L., Sebe, N.: Active domain adaptation with noisy labels for multimedia analysis. World Wide Web **19**(2), 199–215 (2016)
30. McCreadie, R., Macdonald, C., Ounis, I.: Incremental update summarization: adaptive sentence selection based on prevalence and novelty. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 301–310. ACM (2014)
31. McDonald, R.: A study of global inference algorithms in multi-document summarization. Adv. Inf. Retriev., 557–564 (2007)
32. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
33. Rao, J., He, H., Zhang, H., Ture, F., Sequiera, R., Mohammed, S., Lin, J.: Integrating lexical and temporal signals in neural ranking models for searching social media streams. arXiv:1707.07792 (2017)
34. Roegiest, A., Tan, L., Lin, J.: Online in-situ interleaved evaluation of real-time push notification systems. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 415–424. ACM (2017)
35. Ross, S., Gordon, G.J., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: International Conference on Artificial Intelligence and Statistics, pp. 627–635 (2011)
36. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: SIGIR, pp. 373–382. ACM (2015)
37. Sonawane, S.S., Kulkarni, P.A.: Graph based representation and analysis of text document: A survey of techniques. Int. J. Comput. Appl., **96**(19) (2014)
38. Song, J., Gao, L., Nie, F., Shen, H.T., Yan, Y., Sebe, N.: Optimized graph learning using partial tags and multiple features for image and video annotation. IEEE Trans. Image. Process. **25**(11), 4999–5011 (2016)
39. Song, J., Gao, L., Li, L., Zhu, X., Sebe, N.: Quantization-based hashing: A general framework for scalable image and video retrieval. Pattern Recognition (2017)
40. Song, J., Zhang, H., Li, X., Gao, L., Wang, M., Hong, R.: Self-supervised video hashing with hierarchical binary auto-encoder. IEEE Trans. Image. Process. **25**(11), 4999–5011 (2018)
41. Tan, L., Roegiest, A., Clarke, C.L.A., Lin, J.: Simple dynamic emission strategies for microblog filtering. In: SIGIR, pp. 1009–1012. ACM (2016)
42. Tan, H., Ziyu, L., Li, W.: Neural network based reinforcement learning for real-time pushing on text stream. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 913–916. ACM (2017)
43. Wang, X., Gao, L., Wang, P., Sun, X., Liu, X.: Two-stream 3d convnet fusion for action recognition in videos with arbitrary size and length. IEEE Transactions on Multimedia (2017)
44. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3-4), 229–256 (1992)
45. Xu, T., Oard, D.W., McNamee, P.: Hltcoe at trec 2013: Temporal summarization. In: TREC (2013)
46. Xu, J., Liu, X., Huo, Z., Deng, C., Nie, F., Huang, H.: Multi-class support vector machine via maximizing multi-class margins. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 3154–3160 (2017)
47. Yang, M., Mei, J., Fei, X., Wenting, T., Lu, Z.: Discovering author interest evolution in topic modeling. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 801–804. ACM (2016)
48. Yu, L., Zhang, W., Wang, J., Seqgan, Y.Y.: Sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2017)
49. Zhao, W., Wei, X., Yang, M., Ye, J., Zhao, Z., Feng, Y., Qiao, Y.: Dual learning for cross-domain image captioning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 29–38. ACM (2017)
50. Zhu, J., Xie, Q., Zheng, K.: An improved early detection method of type-2 diabetes mellitus using multiple classifier system. Inform. Sci. **292**, 1–14 (2015)
51. Zhu, J., Xie, Q., Wong, W.H., Wong, W.H.: Exploiting link structure for Web page genre identification. Data. Min. Knowl. Disc. **30**(3), 550–575 (2016)
52. Zhu, X., Suk, H.-I., Lee, S.-W., Shen, D.: Subspace regularized sparse multitask learning for multiclass neurodegenerative disease identification. IEEE Trans. Biomed. Eng. **63**(3), 607–618 (2016)