A Multi-task Learning Framework for Abstractive Text Summarization (Supplementary File)

Abstract

In this supplemental file, details including methodology and implementation are provided. We also perform thorough study of the experiment results with quantitative, ablation and qualitative analysis.

Methodology

Assume each input article $X = \{x_1, x_2, ..., x_n\}$ has a corresponding reference summary $Y = \{y_1, y_2, ..., y_k\}$ and a category label L, where n and k denote the length of the input article and the reference summary, respectively. Given an input article X, the abstractive summarization task tries to generate a summary $\hat{Y} = \{\hat{y_1}, \hat{y_2}, ..., \hat{y_m}\}$, where m denote the length of the generated summary. For the text categorization task, given an input article X, our objective is to predict the category label $\hat{L}$ for the input text. For syntax labeling, we have $Z = \{z_1, z_2, ..., z_m\}$ denoting the CCG supertag sequence for the corresponding summary $\hat{Y}$ of source text X.

Shared Bi-LSTM encoder

The abstractive summarizer shares its encoder with the text categorization task. Each word x in the source text is mapped to a low-dimensional embedding $v \in \mathbb{R}^d$ through a word embedding layer, where d denotes the size of the embedding. Then, we employ a bi-directional LSTM (Bi-LSTM) (**?**) encoder to obtain the hidden states of the source text. Formally, given the input word embedding v_i at time step i, the hidden state h_i can be updated with the previous hidden state h_{i-1} as $h_i = \text{Bi-LSTM}(h_{i-1}, v_i)$. In this way, we obtain the hidden states $H = \{h_1, h_2, ..., h_n\}$ for the source text X.

For text categorization task, we use the last item in the hidden vector (i.e., h_n) as the representation of the source text. The text representation h_n is then fed into a task-specific fully-connected layer followed by a *softmax* layer to predict the probability distribution over classes.

Shared LSTM decoder

LSTM decoder is shared by the abstractive summarization task and the syntax labeling task, which is essentially a language model for estimating the contextual probability of the next word except that it is conditioned on the output of the encoder. We use the last item in encoded vector (i.e., h_n) as the initial state of LSTM decoder. On each decoding step t, the decoder receives the input u_t (while training, u_t is the embedding of previous word of the reference summary; at test time it is the embedding of previous word emitted by the decoder) and update its hidden state s_t as

$$s_t = \text{LSTM}(s_{t-1}, c_t, u_t) \tag{1}$$

where c_t is the context vector at time step t. It can be computed as a weighted sum of the hidden states of the encoded input representation H. Formally, we use the attention mechanism (**?**) to calculate the attention weights a_t and the context vector c_t as

$$e_{t,i} = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \tag{2}$$

$$a_{t,i} = \text{softmax}(e_{t,i}), \quad c_t = \sum_{i=1}^{n} a_{t,i} h_i \tag{3}$$

where W_h, W_s and b_{attn} are learnable parameters.

The context vector c_t can be viewed as the representation of the source text at time step t. We then concatenate the context vector c_t and the decoder hidden state s_t at time step t and feed it to a linear function to produce the hidden vector of the decoder:

$$O_t = V[s_t, c_t] + b \tag{4}$$

The generation probabilities of the t^{th} word and CCG supertag can be written as the following,

$$P_t^{\text{w}} = P(y_t | Y_{1:t-1}; X) = \text{softmax}(U^w O_t + b^w) \tag{5}$$

$$P_t^{\text{s}} = P(z_t | Y_{1:t-1}; X) = \text{softmax}(U^s O_t + b^s) \tag{6}$$

where the U^w, U^s, b^w, b^s are parameters to be learned. The superscripts s and w denote supertag and word, respectively. $y_{1:t-1}$ denotes the previously generated tokens. Note that P_t^w denotes the distribution over the whole vocabulary.

However, the pure generation model sometimes suffers from the out-of-vocabulary generation issue and produces many "UNK" tokens in the summary. To alleviate this limitation, copy mechanism is widely adopted in recent abstractive summarization systems (**?**; **?**; **?**). Similar to the work (**?**), in this study the generation probability $p_{\text{gen}} \in [0, 1]$ for time step t is calculated from the context vector c_t, the decoder state s_t, and the decoder input u_t:

$$P_{\text{gen}} = \sigma(V_c^T c_t + V_s^T s_t + V_u^T u_t + b_{gen}) \tag{7}$$

where vectors V_c, V_s, V_u and scalar b_{gen} are learnable parameters.

For each step t, given a candidate token w_j (j denotes the index of the vocabulary), if w_j is out-of-vocabulary token,

then $p_t^w(w_j) = 0$, if it doesn't appear in the source text, then $a_t^j = 0$.

$$P(w_j) = P_{\text{gen}} * P_t^w(w_j) + (1 - P_{\text{gen}}) * \sum a_t^j \quad (8)$$

Multi-task Learning

Our model consists of three subtasks, each has its own training objective. For text categorization task, the objective is to minimize its cross entropy loss.

$$J_{\text{ml}}^{\text{text}}(\theta) = -\sum_{i=1}^{N} L_i log(\hat{L}_i) \quad (9)$$

where θ is the set of parameters, $\hat{L}_i$ denotes the predicted probability distribution over categories.

For the syntax labeling and summarization generation subtasks, we employ the minimum negative log-likelihood estimation:

$$J_{\text{ml}}^{\text{sum}}(\theta) = -\sum^{T} \log(P_t^w) \qquad J_{\text{ml}}^{\text{syn}}(\theta) = -\sum^{T} \log(P_t^s) \quad (10)$$

For the purpose of improving shared Bi-LSTM encoder and LSTM decoder, we train these three related tasks simultaneously. The joint multi-task objective function is minimized by:

$$J_{\text{ml}}(\theta) = \lambda_1 J_{\text{ml}}^{\text{text}}(\theta) + \lambda_2 J_{\text{ml}}^{\text{sum}}(\theta) + \lambda_3 J_{\text{ml}}^{\text{syn}}(\theta) \quad (11)$$

where λ_1, λ_2 and λ_3 are hyper-parameters that determines the weights of the three objectives. Here, we set $\lambda_1 = \lambda_2 = 0.45$, and $\lambda_3 = 0.1$. The parameters for multitask learning are determined by performing grid search on a validation set.

To maximize long-term rewards and alleviate the exposure bias problem (**?**) when generating summaries, we also optimize directly for ROUGE-1 since it achieves best results among the alternatives such as METEOR (**?**) and BLEU (**?**), by using policy gradient algorithm, and minimize the negative expected rewards:

$$J_{\text{rl}}^{\text{sum}}(\theta) = (r(\hat{y}) - r(y^s)) \sum^{n} \log p(y_t^s | Y_{1:t-1}^s; X) \quad (12)$$

where $r(\hat{y})$ is the reward of a greedy decoding generated sequence and $r(y^s)$ is the reward of sequence generated by sample among the vocabulary at each step.

After pre-training the proposed model by minimizing the joint ML objective Eq. 11, we switch the model to further minimize a mixed training objective, integrating the reinforcement learning objective $J_{\text{rl}}^{\text{sum}}(\theta)$ with the original multi-task loss $J_{\text{ml}}(\theta)$:

$$J_{\text{mixed}}(\theta) = \beta J_{\text{ml}}(\theta) + (1 - \beta) J_{\text{rl}}^{\text{sum}}(\theta) \quad (13)$$

where β is a hyper-parameter, and we set β=0.1.

Experimental Setup

Dataset

We evaluate our model on the CNN/Daily Mail Corpus (**?**), which is widely used in abstractive summarization. The dataset comprises news stories in CNN/Daily Mail websites paired with multi-sentences human generated summaries. It consists of 287,226/ 13,368/ 11,490 training/ validation/ test pairs.

For the text categorization task, we explore the source webpage of each news story, which indicates the specific category of each story and provides a label for each story. This dataset covers 11 categories: Sports, Showbiz, Politics, Opinion, Tech, Travel, Health, Crime, Justice, Living and Business.

For our syntax annotation, the training data is annotated with CCG supertags[1], where each word has a corresponding dependency label of supertags.

Baseline Models

Our experiments compare our model with several state-of-the-art methods including the abstractive model (ABS) (**?**), Lead-3 and SummaRuNNer (RuNNer) (**?**), the pointer-generator coverage networks (PGC) (**?**), the abstractive deep reinforced model (DeepRL) (**?**), and a generative adversarial network for abstractive summarization (GANsum) (**?**). Our experiments use the same parameter settings as in the original papers.

Implementation Details

Following the settings of (**?**), we use the non-anonymized version and truncate the input articles/ target summaries to maximum length of 400/ 100 words. We use a 128-dimensional randomly initialized word embedding in our model. Our Bi-LSTM encoder and LSTM decoder have hidden state size of 512 and 256, respectively. We first pretrain ML model for summarization with a learning rate 0.15 (**?**). Then switch to *MATS* training using the Adam optimizer (**?**), with mini-batch of size 16 and a learning rate 0.001. We use beam search with beam size of 5 during decoding.

Experimental Results

Here we compare our model with baseline methods, both quantitatively and qualitatively following the experimental setup of (**?**; **?**).

Methods	ROUGE			Human Score
	1	2	L	
ABS	35.46	13.3	32.65	1.10
Lead-3*	39.2	15.7	35.5	-
RuNNer*	39.6	16.2	35.3	-
PGC	39.53	17.28	36.38	3.38
DeepRL	39.87	15.82	36.90	4.48
GANsum	39.92	17.65	36.71	4.7
MATS w/o text	40.37	17.79	36.75	4.52
MATS w/o syntax	40.62	17.96	37.01	4.64
Our model (MATS)	**40.74**	**18.14**	**37.15**	**5.18**

Table 1: Quantitative evaluation results (ROUGE-N F1 scores on the test set). Those marked with * are extractive models. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script.

Quantitative Evaluation

ROUGE-N is a standard evaluation metric for summarization tasks. It measures the consistency between n-gram occurrences in the generated and reference summaries. ROUGE-L compares the longest common sequence between the reference summary and the generated summary. Table 1 shows the ROUGE-N scores of both our model and the baseline methods, on CNN/Daily Mail corpus. Our model substantially outperforms the baseline methods by a noticeable margin.

[1] https://github.com/uwnlp/EasySRL

Input: University of Waterloo astrophysicists have created a 3D master map of the universe spanning nearly two billion light years. The innovative spherical map of galaxy superclusters is the most complete picture of our cosmic neighbourhood to date. It will help astrophysicists understand how matter is distributed in the universe and provide key insights into dark matter one of physics' greatest mysteries. Scroll down for video a slice through the Map of the nearby Universe. Our Milky Way Galaxy galaxy is in the centre, marked by a cross.(...)
Ground-truth: Map spans nearly two billion light years. Will help astrophysicists predict the universe's expansion. Could help identify where, and how much dark matter exists.
DeepRL: University of Waterloo created a 3D master map of galaxy universe spanning nearly two billion light years. The innovative spherical map galaxy superclusters is the most complete picture of our cosmic neighbourhood to date. It will help astrophysicists understand how matter is distributed in the universe and provide key insights into dark matter. The lighter blue and white areas on the map represent greater concentrations of galaxies.
MATS: Innovative spherical map of galaxy superclusters is the most complete picture of the universe spanning nearly two billion light years. It will help astrophysicists understand how matter is distributed in the universe and provide key insights into dark matter.

Table 2: Example summaries.

Ablation Study

To investigate the effect of each component of the MATS model, we also perform the ablation test of MATS in terms of discarding text categorization (w/o text) and syntax generation (w/o syntax), respectively. The results are reported in Table 1. Generally, both tasks contribute, and text categorization contribute more. This is within our expectation since the the text categorization helps learn better category-aware representations. While the improvement of integrating syntax generation is relatively limited. This may be because that the issue of the incomplete sentence has little effect on the evaluation metrics of summarization.

Qualitative Evaluation

We also perform human evaluation to evaluate the informativeness and fluency of the generated summaries by randomly select 50 examples from the test set. Similiar to Cheng et al. (**?**), two human evaluators were invited to rank each summary generated by all models except Lead-3 and RuNNer[2] based on their informativeness (if the summary captures important information in the article) and fluency (if the summary is written in well-formed English). For the informativeness part, the human evaluator will read through the summaries to try to get a sense of what the story is talking about. If the human evaluator does not get an idea of what the original story is talking about, then this summary will be ranked lower. For the fluency part, if the summary is not well-written (e.g., grammatical mistakes), it will have a lower ranking score. The human evaluators are required to perform ranking of summaries by taking the above 2 factors into consideration, where 1 indicates the lowest level and 7 indicates the highest level. The experimental results based on human evaluation are summarized in Table 1. *MATS* achieves the best results.

To evaluate the proposed model qualitatively, we report some generated summaries by different models. Due to limited space, we randomly choose one generated summary by DeepRL and our model from test data for comparison. The results are reported in Table 2. We observe that *MATS* tends to generate more specific and meaningful summaries given the source articles. For example, our model successfully catches the key point of "dark matter" and "the size of map", while DeepRL attends over some trivial facts except the key facts.

[2]The two models are not directly comparable to ours as they are based on an anonymized version of the dataset