

Monotonic alignments for summarization[☆]

Tonglee Chung^a, Yongbin Liu^{b,*}, Bin Xu^a

^a Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

^b Computer School, University of South China, Hengyang 421001, China

ARTICLE INFO

Article history:

Received 10 January 2019

Received in revised form 1 October 2019

Accepted 8 December 2019

Available online 14 December 2019

Keywords:

Summarization

Monotonic

Alignment

Attention

ABSTRACT

Summarization is the task that creates a summary with the major points of the original document. Deep learning plays an important role in both abstractive and extractive summary generations. While a number of models show that combining the two gives good results, this paper focuses on a pure abstractive method to generate good summaries. Our model is a stacked RNN network with a monotonic alignment mechanism. Monotonic alignment has an advantage because it produces the context that is in the same sequence as the original document, at the same time eliminating repeating sequences. To obtain monotonic alignment, this paper proposes two energies that are calculated using only the previous alignment state. We use sub-word method to reduce the rate of producing OOVs (Out of Vocabulary). The dropout is used for generalization and the residual connection to overcome gradient vanishing. We experiment on CNN/daily new and Reddits dataset. Our method outperforms the previous models with monotonic alignment by 4 ROUGE-1 points and achieves the results comparable to state of the art.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Summarization is the task of producing a shorter piece of text that resemblances the main idea of a long document. Current approaches can be categorized into abstractive and extractive, where the former choose words based on generating probabilities [1] and the latter relies on copying or pointing to sentences [2]. With the development of encoder-decoder, automatic summarization has greatly improved by solving some fundamental problems, e.g. correctness, repetition, OOV (Out of Vocabulary). In this paper, we design an alternate monotonic attention for summarization. Our model relies on stacked RNNs for generating correct content, monotonic alignment for repetition and sub-word for OOV. We also investigate a different challenge where contents of summary and document are not in the same order, especially for multi-sentences summarization. Consider reading a summary where the story line of events is different from the original News article, this would lead to confusion thus lowering reading experience. We also investigate the problem where the model skips a lot of content. Hence this paper proposes a model where order of content is guaranteed to be in the same order by

using progressive energy and constraining coverage of summary by using distance energy.

Summarization models based encoder-decoder with attention have been very effective. Both abstractive and extractive models have been greatly improved and a number of current researches focus on merging the two. There are a few components that are vital in a number of models that produce SOTA (state of the art) results: 1. pointer mechanism to produce accurate words and remove OOV; 2. coverage to reduce probability of repetition [1]; 3. reinforcement learning objective to improve ROUGE score [3]; 4. sentence attention that chooses sentences with highest ROUGE score [4]. The pointer mechanism calculates a probability of using a word that is attended to or using a word that is generated. Studies show that most of the time, the pointing mechanism is triggered thus resulting mostly using words from the original document. Abstractive has more potentials of exceeding one's expectation and it is not pushed to its limit, so it is important to conduct research in areas using pure generative probabilities to predicts words in the summary. While the coverage mechanism shows great potential in reducing repetition, it is vital that we have the different alternative to solve one problem. Reinforcement learning has given the power to directly maximize the evaluation metric, but some researches [3] show that high evaluation scores do not result in good readable summary. This may be caused by the evaluation metric itself, where having the longest common sequence does not yield readable text, so it is necessary to conduct research by using fundamental cross entropy loss.

[☆] No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to <https://doi.org/10.1016/j.knosys.2019.105363>.

* Corresponding author.

E-mail addresses: tongleechung86@gmail.com (T. Chung), yongbinliu03@gmail.com (Y. Liu), xubin@tsinghua.edu.cn (B. Xu).

Due to the above importance, we present a new monotonic alignment mechanism for summarization and propose the alternate methods to some fundamental problems. Key points include: 1. using monotonic attention to prevent repetition and keeping contents in the same order, 2. relying entirely on the generative probability of decoder to produce summary without any pointing/copying mechanism, and 3. training solely on entropy based loss with some constraint specific to problems. We use a stacked RNNs as encoder and decoder. Such a method is used in machine translation and is proven to be very effective. The stacked architecture learns representation from prior layers to produce a higher level of abstraction. The decoder includes a monotonic hard alignment, which reduces repetition and produces alignments that are monotonically in order. We use dropout layer in the RNN cells for regularization and residual layers between each RNN layers to prevent gradient vanishing. We use sub-words to reduce the probability of OOV occurring in the training and inference stage.

The main contributions of our work include:

- A monotonic alignment mechanism which considers both monotonicity and distance of sequence at the same time, shows its effectiveness in summarization task.
- We propose two energies and their corresponding training criteria for monotonic alignment calculation and show their effectiveness.
- Our method based monotonic alignment achieves state of the art results on CNN/Daily News and Reddits dataset. Our model does not use any ROUGE maximizing objective to achieve good results.

The rest of the paper will be organized as below. The following section will introduce related works. Our monotonic alignment will be described in Section 3, followed by an overview of the entire summarization model. Experiments and results will be presented in Section 5 and the paper will be concluded in the last section.

2. Related work

Applying the encoder-decoder attention framework to the task of summarization has attracted a lot of attention. In abstractive summarization, [5] and [6] were some of the first pioneers in the research area when they directly utilized the framework and achieved inspiring results. A great breakthrough was the development of pointer/copy mechanism to the framework to address the issue of incorrect context and OOV; and the introduction of coverage to handle repetition problem when decoding [1]. And another boost was when Reinforcement Learning was utilized [3], which allowed ROUGE score to be optimized directly, the research shows that combining entropy loss with the reinforcement learning objective yields readable results with high scores. Extractive summarization has also achieved impressive results by applying the same framework but changing to sentence classification task [2], the training objective is switched to choosing the sentences that result in highest scores, we can think of this as indirectly optimizing the ROUGE score.

Recently emerged researches focus on merging the two worlds of abstractive and extractive, mainly by choosing sentences that are most informative, then producing summary over the chosen sentences [7,8]. In these researches, the models encode at a different level of the input sentence, usually words and sentences. Applying reinforce learning to the unified model has also greatly improved summary results [4,9,10], these researches apply reinforcement learning to train different parts of the model, sentence extractor, the output sequence, etc. Some branches reach out to using generative models for summarization [11,12].

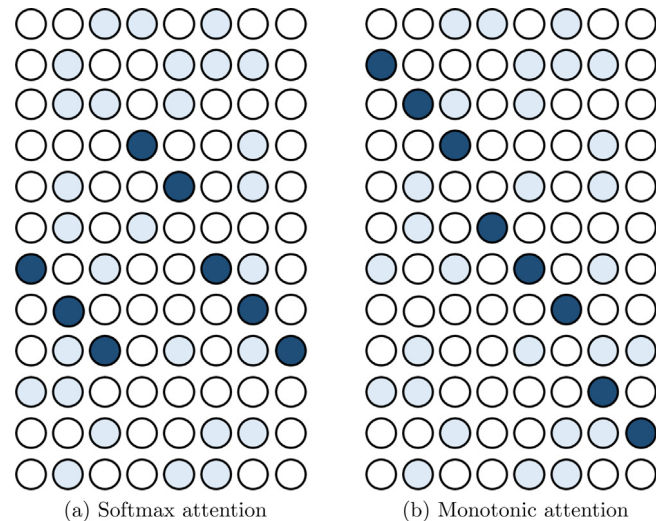


Fig. 1. On the left is a softmax based attention, elements do not follow any order and areas may be attended to multiple times, whereas in a monotonic setting, attended elements follow a order and does not repeat.

Other researches [13–17] include being faithful by extracting actual fact descriptions [18], where the problem of fake summaries was addressed; incorporating key information to the model [19], where keywords are extracted and later used as long term guide; using global context [20] during the decoding process; producing customized summaries [21] according to personal preferences.

There have been researching papers that present their findings in using monotonic alignment in the task of summarization [22, 23], while they show that it shows potential in single sentence summarization, it still struggles when trying to do multi-sentence summarization.

3. Monotonic alignment

Attention mechanism is crucial in a encoder decoder model. Most of the time, it is speculated that most of what is generated during summarization is what is attended to (gaining the largest attention weight). This is also true for other sequence tasks such as machine translation and sequence tagging. This aligning mechanism has given such great success to sequence task in NLP. In the task of automatic summarization, we find that most of the time, attended tokens are generated. So theoretically, by manipulating the way attention weights are calculated, we could get summaries that are desired. Typically, during each decoding step, a score (energy) is calculated for each token in the input sequence (the original document), then a softmax function is applied to get a probability distribution. In a traditional softmax attention calculation, there is limited access to previous information, especially after decoding some step, information is dispersed, thus the problems arise in summary generated. Information of what is previously produced may have little impact on latter outputs, thus repetition occurs, as well as summaries which are not in order. Fig. 1 shows a figure of general attention alignment and a monotonic alignment. In a general attention probability distribution (Fig. 1(a)), there is no order in the part that is attended to, and as more text gets decoded, repetition occurs as well. Monotonic attention (Fig. 1(b)) on the other hand displays monotonicity in the attention distribution, where attended tokens follow a descending order. In other words, attended words will be to the right of previously attended tokens.

We propose an effective way of calculating monotonic attention probability in this section. We define two energy function that can be calculated using only the previous attention distribution. The calculation is simple and effective, with only two sets of extra weights to train. The first energy is called progressive energy, which makes sure that attention probability is descending. During decoding with monotonic attention, a large portion of content may be skipped due to the nature of the attention, so another energy is introduced to encounter this. Distance energy ensures that the next attention is not too far from the current attended token.

Notation wise, bold lower case is used to denote vectors (e.g. $\mathbf{v}$, $\mathbf{h}_i$), bold upper case for matrices (e.g. $\mathbf{W}$, $\mathbf{U}$), cursive upper case for sets (e.g. $\mathcal{V}$), capital letters for represent sequences (e.g. X , Y) and lower case for individual elements in the sequence (e.g. x , y). Throughout the paper, subscript is used to denote the time-step and superscript for layers (s_i^l time step i at layer l) unless otherwise stated.

Next, we introduce a typical soft alignment method and present our monotonic alignment with progressive energy and distance energy.

3.1. Soft alignment

Attention mechanism gives a probability distribution for the decoder to access the memory from the encoder (hidden states of each element in the original document). A core component of the probability distribution calculation is the energy function, which is the function to compute an energy score the token in the input sequence. There are several functions that are introduced, of which Bahdanau Attention and Luong Attention are the two more popular ones.

Attention distribution calculation comes in the following form, for each token of the input sequence, the energy function takes in the current state vector and the vectorized memory representation and output a scalar value. Softmax is then performed over scores for each token, as shown below:

$$\begin{aligned} e_{i,j} &= \text{EnergyFunc}(\mathbf{s}_i, \mathbf{h}_j) \\ \alpha_{i,j} &= \text{softmax}(\mathbf{e}_{i,:})_j \end{aligned} \quad (1)$$

$\mathbf{e}_{i,j}$ represents the score for element j at time-step i , it is calculated using the base LSTM state at time i and memory state of element j .

Energy function comes in many forms, of which Bahdanau and Luong are very common. Bahdanau function uses a linear function to combine the two input vector $\mathbf{s}_i$ and $\mathbf{h}$ as shown below:

$$\text{BahdanuEnergyFunc}(\mathbf{s}_i, \mathbf{h}_j) = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{s}_i + \mathbf{V}\mathbf{h}) \quad (2)$$

then the output is activated with a tanh non-linearity. Finally, the output is multiplied with a weight of size `[hidden_size, 1]` to get the scalar output. Usually, the input vectors $\mathbf{s}_i$ and $\mathbf{h}$ are of size `hidden_size` and it is convenient to set weights $\mathbf{W}$ and $\mathbf{V}$ to size `[hidden_size, hidden_size]`. This approach introduces two new sets of weights to train, and Luong attention uses dot product instead of the linear function to merge the two input vector as shown below:

$$\text{LuongEnergyFunc}(\mathbf{s}_i, \mathbf{h}_j) = \mathbf{s}_i \cdot \mathbf{h}^\top \quad (3)$$

the dot product of two vectors with the same dimension size will produce a scalar value which is then used as the energy score in the attention mechanism. Luong attention does not introduce new weights to train so the information is contained within the representation. Though Luong attention is claimed to have better performance in machine translation, the same results do not apply to automatic summarization, so in our experiment, Bahdanau attention is used for better performance.

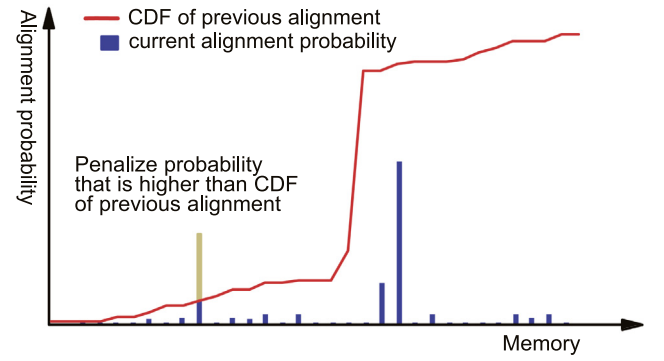


Fig. 2. Visualizing the objective of progressive energy. Alignment probabilities that exceed the CDF of previous alignment is penalized thus constraining elements to the left not to be focused on. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Monotonic alignment

Early attempts to generate attention are very straightforward. A hard constraint was put on the attention probability distribution where the token with the highest score is assigned with a 1 probability and the rest with 0. During the next decoding step, a mask is placed over the memory to calculate only scores that are to the right of the previous token with 1 probability. Though able to produce monotonicity, the proposal is a hard attention where only one token gains all probability. But there is a strong assumption that the input and output sequences follow a strict monotonic order. We improve this by removing the hard constraint of only choosing to the left of the currently attended token. During training, a hard softmax is performed on the memory observed to obtain a point to continue from, which means that the output of the current token solely depends on the memory of that particular token. In our approach, we hope to soften that hard constraint, so that even the current output is generated with an aggregation of the whole memory.

To achieve such monotonic attention, we propose using the accumulated sum of previous attention distribution over the input sequence to penalize generation process such that it follows a monotonic formation. This is called a progressive energy because energy progresses to the left. Because attention distribution is monotonic, which does not attend to token to the right, if the attention weight focuses too far from the current token, we find that large portions of the document are skipped, we introduce another energy to encounter this effect. The distance energy is a smooth energy function that ensures that the mass of energy is focused around what is currently attended to.

3.2.1. Progressive energy

To enforce monotonicity, we propose using a cumulative vector of the previous attention distribution as energy to constrain generation of current attention distribution calculation. The basic idea of the proposed energy is that we need to find a point where attention gathers, and find a penalizing term to constrain the attention distribution from attending to memories to the right. One such function that satisfies such condition is the cumulative sum of the previous attention distribution. Fig. 2 shows a demonstration of cumulative energy (red line). The previous energy (blue histogram) shows that model what part was previously attended to, the cumulative sum of that gives a nice discrete line that sums to 1 when getting to the end (attention probability distribution sums to 1). Each token (memory vector) would have

a value that is the cumulative sum of all attention weights to the right.

Ideally, we want to see the attention weights generated be less than the cumulative weight of the previous attention probability distribution. As mentioned previously, we called this weight the progressive weight for convenience. Progressive energy for element i given a alignment distribution α can be formulated as:

$$\begin{aligned} \text{Progressive}(\alpha, i) &= \text{CumSum}(\alpha, i) \\ &= \{\alpha_1, \alpha_1 + \alpha_2, \dots, \alpha_1 + \alpha_2 + \dots + \alpha_i\} \end{aligned} \quad (4)$$

CumSum corresponds to cumulative summation, where the value at i is basically the sum to previous i elements. This is similar to coverage approach in some sense such that coverage accumulates over each step, but it is obvious that an extra vector needs to be maintained throughout the whole decoding process, while the progressive energy only requires previous attention probability distribution. Another difference is that, coverage only addresses the repetition problem, and prevents token to not gain attention if previously attended to, with no assurance of monotonicity. Also note that since monotonicity ensures that the large mass of attention comes from tokens that are to the left of previous point of mass, repetition is eliminated naturally.

The choice of using cumulative attention is for its simplicity. The function is provided by many tools and libraries¹ and as mentioned before there is no need to maintain an extra vector, all that is needed is the previous attention probability distribution. The proposed energy also has the property of being ranged between $[0, 1]$.

3.2.2. Distance energy

Similar to progressive energy, we hope that the distance energy can give constraint to the scoring process such that the alignment is penalized when the next word is too far from the previously attended element (see Fig. 3 for visualization). To obtain such energy, we propose using a weighted sum of the previous alignment where weights decrease according to distance. A normal distribution function is utilized as the weights in our implementation, we normalize it to be between $[0, 1]$. The normal distribution is smooth and where values decay at both sides of the mean. The distance energy is a vector with the same size as alignments (size of original document), where for each element i , the weight value of element j (notated as $p(i, j, \sigma)$) is calculated by the value of a normalize normal distribution with mean i and variance σ at j . The distance energy value for element i is the sum of weight alignment as shown below:

$$\begin{aligned} \text{Distance}(\alpha, i, \sigma) &= \sum_j p(i, j, \sigma) \times \alpha_j \\ p(i, j, \sigma) &= \exp \left(-\frac{(j-i)^2}{2\sigma^2} \right) \end{aligned} \quad (5)$$

distance energy values are ranged between $[0, 1]$ because the sum of alignment values is strictly equal 1, and the weight value is ranges between $[0, 1]$. Weights are calculated using the index, in other words, it is based on the position (the further away two elements the smaller the weight).

3.2.3. Monotonic energy function

The monotonic alignment energy function (or score function) is very similar to the typical way with the only difference being the progressive energy and distance energy being calculated first using the previous alignment distribution. Then a score for each

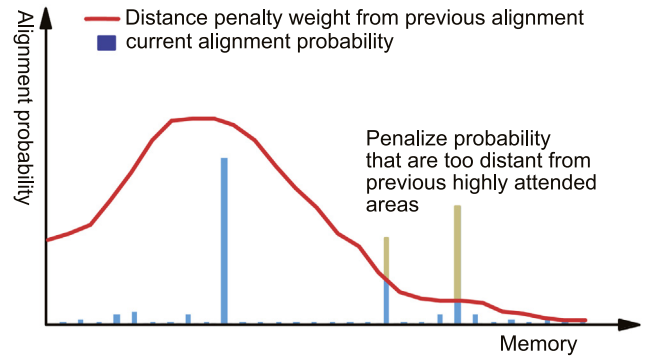


Fig. 3. Visualizing the objective of distance energy.

element in the memory is calculated using a slightly modified energy function. Finally, it is normalized to a probability distribution using a softmax function. The process can be formalized as:

$$\begin{aligned} \mathbf{f}_j &= \text{Progressive}(\alpha^{i-1}, j) \\ \mathbf{d}_j &= \text{Distance}(\alpha^{i-1}, j, \sigma) \\ e_{i,j} &= \text{EnergyMono}(\mathbf{s}_i, \mathbf{h}_j, \mathbf{f}_j, \mathbf{d}_j) \\ \alpha_{i,j} &= \text{softmax}(\mathbf{e}_{i,:})_j \end{aligned} \quad (6)$$

The energy function is formalized in Eq. (7). The two energies are fed through an unbiased linear function and add to the processed state and memory vector.

$$\begin{aligned} \text{EnergyMono}(\mathbf{s}_i, \mathbf{h}_j, \mathbf{f}_j, \mathbf{d}_j) \\ = v^T \tanh(W\mathbf{s}_i + V\mathbf{h}_j + U\mathbf{f}_{i,j} + G\mathbf{d}_{i,j}) \end{aligned} \quad (7)$$

Two extra weights $\mathbf{U}$ and $\mathbf{G}$ are included to accompany the transformation of the two energies.

3.2.4. Penalizing loss

To learn the weights of the constraining energy, we need to include a penalizing term for each energy. As we mentioned in our objective of the energies, the aligning score needs to be penalized when exceeding the energy value. To obtain penalizing loss, we do the following operation, get the max of the current aligning score and energy score, then subtract the energy score as shown in the equation below:

$$\begin{aligned} L_{pro} &= \sum_j \max(\alpha_{i,j}, f_{i,:}) - f_{i,:} \\ L_{dis} &= \sum_j \max(\alpha_{i,j}, d_{i,:}) - d_{i,:} \end{aligned} \quad (8)$$

The final loss function is the sum of entropy based loss and the penalizing terms.

$$L_{final} = L + L_{pro} + L_{dis} \quad (9)$$

4. Stacked encoder decoder model with monotonic attention

Our model is a stacked RNN encoder decoder with one monotonic attention mechanism layer as shown in Fig. 4. It consists of two major components: encoder and decoder. The encoder takes in as input the embedding of the original document and outputs one vector representation of each word, this list of vectors is also called the memory in attention mechanism. The decoder takes in the memory and previous output to generate the next word based on an attention mechanism, the process is stopped when the length limit is reached or an end of sentence symbol is produced. The attention mechanism is the process of evaluating the previous output and computing the probability distribution of which regions in the original document to focus on in terms of

¹ Provided in Tensorflow https://www.tensorflow.org/api_docs/python/tf/cumsum.

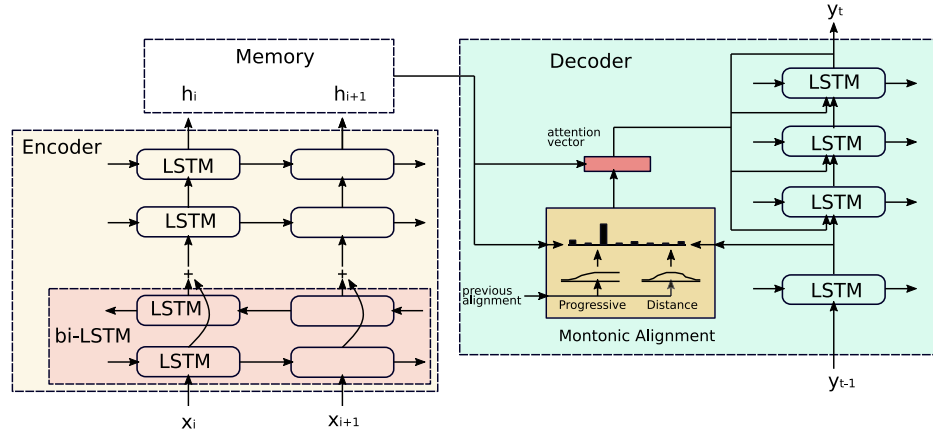


Fig. 4. The stacked RNN model with multiple layers for summarization. For clarity, we remove embedding layer for input; projection and softmax layer for prediction. Only stacking of four RNNs is shown as a demonstration. The original document is encoded using a stacked LSTM which has a bi-LSTM cell at the base. The decoder, also a stacked RNN, has one attention layers at its base, a monotonic aligning mechanism is used, it ensures the desired order of attention.

weight. Our model is adapted from GNMT [24], a system that has been successful in the field of machine translation.

The problem of summarization is defined as follows: given a sequence of words $X = x_1, x_2, \dots, x_m$ of length m , output a shorter sequence $Y = y_1, y_2, \dots, y_n$ with n words. All words in both input and target sequence come from the same vocabulary $\mathcal{V}$. During decoding, the task boils down to predict the next word given the original sequence and previously decoded words:

$$P(\mathbf{Y}) = \prod_{t=1}^n p(y_t | y_1, \dots, y_{t-1}, \mathbf{X}) \quad (10)$$

As shown in the equation in the above equation, the probability of the summary is the product of the probability of each token in the generated summary given the original document and previously generated tokens. The stacked encoder decoder framework is initially the Google Neural Machine Translation system and modified for summarization. The system displays some advance traits such as training parallelism, dropout, residual connection, and so on.

4.1. Stacked RNN encoder

The encoder is a stacked RNN encoder which takes in the original document token sequence $\mathbf{X} = x_1, x_2, \dots, x_n$ and projects each token to a hidden representation space, which is later used in the decoder as memory. The output for the encoder is a fixed length vector for each token $\mathbf{H} = h_1, h_2, \dots, h_n$. The projection involves stacking RNN networks with LSTM cells at its core.

Input tokens are projected into an embedding space as most deep learning NLP does. The embedding table $\mathbf{E}$, with dimension $vocab_size \times hidden_dim$, is randomly initialized with each token in the vocabulary dictionary represented by a fixed length vector. These vectors are trained and updated as parameters of the model. A lookup table is used to convert the one-hot input sequence into a vector representation of the input document. Then the embedding is fed into a stacked RNN to produce the final hidden representation as follows:

$$\begin{aligned} \mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m &= \text{embedding_lookup}(\mathbf{E}, \mathbf{Y}) \\ \mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_m &= \text{StackedRNN}([\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m]) \end{aligned} \quad (11)$$

As shown in Fig. 4, the first layer of the stacked RNN is a Bi-LSTM network. It takes in the embedding representation of the input tokens and outputs a hidden representation. While the uni-directional LSTM captures information from right to left, the Bi-LSTM runs LSTM in two directions (left to right and right

to left). Many researches have found that this structure is able to outperform the former and has become the go-to structure for NLP task. The output is the concatenation of the two LSTM network. We denote this as:

$$\mathbf{h}_1^1, \mathbf{h}_2^1, \dots = \text{Bi-LSTM}(\mathbf{e}_1, \mathbf{e}_2, \dots) \quad (12)$$

where the superscript denotes the layer and the subscript denotes the position of the representation $\mathbf{h}_1^1$ would mean the representation for the first token in the first layer. This output is used as the input for the next RNN layer.

Multiple layers are stacked on the 0th layer, with the l th layer taking in the output from the previous layer $\mathbf{h}^{l-1}$ and produce the output for the current layer as shown in Eq. (13). The final output of the stacked RNN is the output of the final layer $\mathbf{h} = \mathbf{h}^n$.

$$\mathbf{h}_1^l, \mathbf{h}_2^l, \dots = \text{LSTM}^l(\mathbf{h}_1^{l-1}, \mathbf{h}_2^{l-1}, \dots) \quad (13)$$

The output for the final layer is used as the output of the encoder which is layer used as the memory for the decoder. Studies have shown that such a structure has provided good results for machine translation. It is able to produce high abstraction of the input document with the deep layers of RNN.

4.2. Stacked decoder

The decoder generates token based on the outputs of the encoder together with previously generated tokens and produces a probability distribution over all tokens in the vocabulary $p(w_t | \mathbf{h}, w_0, w_1, \dots, w_{t-1})$. The probability function is a complex non-linearity that involves a stacked RNN structure which attention mechanism. In this section, we introduce the monotonic attention mechanism which mainly involves work in the decoder.

The decoder is a stacked LSTM as well as depicted in Fig. 4, but differently structured. The first layer is an RNN with an LSTM cell, which takes in the previous output token and previous cell state to emit a new cell state and hidden output. This layer also computes attention distribution for the memory [25]. In our experience, we find that because it is a deciding factor in the quality of the summary. Usually, the attention distribution is greatly skewed, where only one or a very little number of tokens representation gain 80 to 90 percent of the distribution weights, and this/these words would dominate how words are produced. While a large fraction of this section focuses on the attention mechanism, we outline the framework here for continuity and consistency. The attention mechanism outputs a context vector that is used to compute the vocabulary distribution $p(w_i)$. To increase the abstraction of the representation, more RNN layers

are stacked on top. These layers take in the output from the bottom layer and context vector from the attention mechanism as shown in Fig. 4.

More formally, the decoder takes in as input the memory from the decoder $\mathbf{h}$ and generates the output summary. Starting from the first RNN layer, during each time step (meaning the generation of each token in the summary), the LSTM cell takes in the previous cell state and the embedding from the previously generated token to compute an output, which is later used in the attention mechanism to calculate attention distribution and context vector.

$$\begin{aligned} \mathbf{s}_t^1, \mathbf{z}_t^1 &= \text{LSTM}^1(\mathbf{s}_{t-1}^1, \mathbf{y}_{t-1}^1) \\ \alpha_i &= \text{alignment}(\mathbf{s}_t^1, \mathbf{h}_i, \alpha^{t-1}) \\ \text{context} &= \sum_i \mathbf{h}_i \times \alpha_i \end{aligned} \quad (14)$$

LSTM^1 refers to the LSTM cell in the first RNN layer. Attention mechanism would use this information to access the memory and compute attention distribution. We formalize this with the alignment function (see Section 3 details of this function). Finally, the context vector is calculated as the weighted sum of the memory. It is important to note that while the last output from the encoder is used as the input for the first step, we are not able to apply such methods here. The structure for the decoder and encoder can be different, the encoder may have more or less layer thus causing inputs to be unbalanced. Also, we hope that there can be more flexibility in the hidden unit size, and that cannot be guaranteed if input size is pre-defined (hidden size needs to be the same as the input size in order to perform matrix multiplication). Instead, we initialize the input to a zero vector, we find that the initial input does not make a huge difference in performance.

For the RNN layers stacked on top, the LSTM cells are also used, but besides taking in the cell output from the lower layer, they also take in the context vector. Another significant difference from the first layer is that these layers do not compute attention. All layers above the first layer use the same context vector from the first layer. In our experiment, attention distribution for every layer were also tested and we find that the distribution are more or less the same, so that we conclude that these layers learn the same distribution. Thus it is not necessary to train a different attention network for each layer.

$$\mathbf{s}_t^l, \mathbf{z}_t^l = \text{LSTM}^l([\mathbf{s}_{t-1}^{l-1}, \text{attention}], \mathbf{z}_{t-1}^l) \quad (15)$$

LSTM^l refers to the LSTM in l th RNN layer. The cells in this layer are basically a non-linear transformation of the context vector and cell output. Through our experiments, we find that the deeper layers have a positive effect on the evaluation performance of the system, but at a certain deep they not only do not improve but degrades. Setting the RNN layer depth to 4 to 8 has similar results.

Generation of the output sequence is a deterministic process, where at each time step a probability distribution is calculated for the tokens in the vocabulary. In general, a greedy approach can be used to choose the token that is to be produced. The probability can be calculated by using the output hidden state of the last decoder layer. Another approach is to use the context vector from the attention mechanism to produce the token distribution, this would allow the output to be closely related to the original. In our approach, we use a combination of both by performing a linear transformation and projecting the combined representation to the token space as shown in the following equation:

$$P_{\text{vocab}} = \text{softmax} \left(\sum V \times [\mathbf{z}_t^l, \text{attention}] + b \right) \quad (16)$$

In the inference stage, the whole output sequence is generated with only the input document. So outputs for each step rely solely

on the words that are generated previously. Training is different from inference such that a reference is provided, and the model is trained based on that reference. Next, we introduce training in detail.

4.3. Training objective

Training a sequence to sequence model involves minimizing the cross entropy loss between the generated tokens and the target summary. Such loss is used very commonly in many other sequence tasks, such as neural network language modeling and neural machine translation. This is used to measure the distance between the generation probability and the target distribution. Readers should note that there is a fundamental difference between training and inference. In inference, the decoder is provided with only tokens that are previously generated, while during training, the decoder is provided with the subsequence prior to the token to be generated. The target token is one hot distribution, which means only one token in the vocabulary is 1 while the rest are 0s. The distance between a generated distribution and target distribution is then minimized. Generally, the cross entropy is the product of the real target and negative log generated probability summed together. But because there is only one token in the vocabulary with a 1, we can simplify that to minimizing the negative log of the generative probability of the target token, as shown below:

$$\begin{aligned} L &= - \sum_t \sum_i y_t^i p(w_t^i) \\ L &= - \sum_t \log p(w^t) \end{aligned} \quad (17)$$

During implementation, a method called the shift right is used where the input of the decoder is the target summary and the target tokens is the same but shifted one to the right. This is for quick implementation and easy to obtain the cross-entropy loss.

Training the model requires calculating gradients and updating the parameters. During our experiments, we find that using the vanilla stochastic gradient descent (SGD) optimizer does not result in the model converging and the model usually cannot produce good results. We find that by using ADAM (Adaptive Moment Estimation) optimizer, the model is trained quicker and it converges.

The final loss function is the sum of entropy based loss and the penalizing terms.

$$L_{\text{final}} = L + L_{\text{pro}} + L_{\text{dis}} \quad (18)$$

4.4. Sub-word inputs

Out-of-vocabulary is a vital issue in many NLP task, which is the problem of having tokens that do not appear in the vocabulary [24,26]. This results in models have bad outputs and not learning the representation for some tokens. This is the result of the long tail problem, where some token with low frequency are discarded. This is due to the size of the vocabulary growing to a size too large to handle and have enough data points to train the representation. Generally, there are three way of addressing the issue. 1. use character level representation and learn a network to compose tokens using the character representations; 2. use a pointing mechanism to copy the unknown word; 3. use a sub-word mechanism which is between the using character level representation and whole token representation. Learning the character level representation greatly reduces the size of vocabulary size but it tries to embed a lot of information into a small number of vectors. It brings the problem of under represented information embedding where the information is easily

lost after a few rounds of parameter update. Pointing probability is calculated telling the model to use the word that is generated by the model or the word that is highest attention probability, but in this approach, the unknown word does not have representation, and knowledge of the token is not stored in memory. In our implementation, we use the sub-word approach to address the OOV problem. The sub-word approach is data driven and generates a segmentation for any possible token.

The process of generating the sub-words is deterministic and is usually trained on a large corpus. This is useful for improving the robustness of the system. This approach inherits the benefit of having very little [UNK] tokens in both the input and the output without increasing any special copy mechanism. The size of the vocabulary can be set before training so it can offer more flexibility depending on the machine specification. Based on frequency of words, rare or unobserved words are broken down to sub-words (also know as word pieces). In the example below, the unobserved word *zvyozdochka* is broken down in five parts: *_z*, *vy*, *oz*, *doch* and *ka*.

- **original:** submarine is in zvyozdochka shipyard
- **sub-word:** *_submarine_is_in_z vy oz doch ka _shipyard*

in the example, if the word tokens are used, [UNK] will very likely be used to represent *zvyozdochka*, because it is such a rare word. But if the sub-word input is used, each token will have representation and will be trained during the training process. Training of the segmentation model is relatively fast, as it takes a few minutes to train a corpus of 500,000 sentences. Conversion from the sub-word form to the original form is also very simple. It can be achieved by simply removing all white spaces in the sequence then replacing all *_* with white spaces.

4.5. Residual layers

Deep stacked RNNs provide more abstraction to the input and often gives better results [24]. However the deepness brings the problem of vanishing and exploding gradient, thus making it slow and difficult to train. Using residual layers to connect the stacked RNN can greatly resolve these matters. The residual connection concatenates the input and output of a cell and gives that as the new output as described with these equations:

$$\begin{aligned} \mathbf{s}_t, \tilde{\mathbf{y}}_t &= \text{LSTM}(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}) \\ \mathbf{s}_t, \mathbf{y}_t &= \mathbf{s}_t, [\tilde{\mathbf{y}}_t, \mathbf{y}_{t-1}] \end{aligned} \quad (19)$$

The residual layers provide a path for gradients to flow back and allow the deeper layer to be trained more efficiently.

4.5.1. Dropouts

Over-fitting is a serious problem for most neural network, which is when the model is over confident in the training set and performs badly in the test set. In this model, dropout is used to address this problem. Dropout can be implemented as simple as randomly selecting a number of weights in the neural network and setting them to 0. In actual implementation, a masking metric consisting of only 0s and 1s is used over the weights. The dropout rate is set in advance which is the percentage of weights that are dropped during training. Dropout is implemented as follows:

$$\begin{aligned} r^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{y}^{(l)} \times r^{(l)} \end{aligned} \quad (20)$$

in the equation, the dropout is to be performed to the weights $\mathbf{y}^{(l)}$. A metric $r^{(l)}$ is sampled from a Bernoulli distribution with a rate of p . During the update of weights in training, because of the mask $r^{(l)}$, gradients with 0 mask will not be passed downward in back propagation. A gradient multiple by 0 will cause the update of the weight to 0 and thus no update will be performed on that weight unit.

Table 1
Statistics for the Reddit dataset.

Data	Number of examples
Training data	358528
Evaluation data	44674
Test data	44488

Table 2

We use pointer-generator as our baseline on CNN dataset. Our results are comparable to the baseline, but have some distance with reinforcing models. ROUGE score is in favor of extractive models and reinforcement learning models because these models directly or indirectly are maximized the score so we do not directly compare with these models. But we still report results from these models. (SummaRuNNer [2])

	Rouge-1	Rouge-2	Rouge-L
Baseline models			
Hard monotonic [17]	31.14	15.76	–
MoChA [18]	35.46	13.55	–
Pointer-generator	39.53	17.28	36.38
Models in this paper			
StackedRNN	33.51	13.06	30.69
StackedRNN+pro	36.89	15.26	33.81
StackedRNN+pro+dis	39.91	17.06	36.24
Extractive + abstractive models			
SummaRuNNer	39.6	16.2	35.3
[7]	40.68	17.97	37.13
Reinforcement learning models			
[10]	41.69	19.47	37.92
[9]	41.25	18.87	37.75

5. Experiments and results

While previous summarization experiments with monotonic alignment conducted on short sentence summary show positive results [22], multi-sentence summarization is still a struggles [23]. In this paper, we use CNN\Daily News and Reddit datasets to evaluate our model. we use Pointer-generator [1] and Raffel [22] with monotonic attention as our baselines. The CNN\Daily News dataset has over 220 thousand training pair and more than 11 thousand for evaluation and testing.

The Reddit dataset [27] consists of around 4 million Reddit articles and a short description of the article which can be used as the summary. A number of the Reddit articles have a common practice of appending a ° TL;DR to the end of the post, which is the acronym of ° Too Long, Didnt Read. The original dataset is rather noisy and some data points are not suitable for the task of summarization, so we perform the following steps to filter the data. 1. Filter out articles that are too long or too short. We view the summarizations of texts that are too short to have the meaning as there are not many points of shortening short text. We set the maximum and minimum word count threshold to 200 and 1000. The texts that are too large are also removed as they hold too much information and may lead too much difficulty in training the model. 2. Filter out articles with target summary that are too long or short. We find that the short context in ° TL;DR is not from the article and is not the summary. The threshold set of maximum and a minimum number of tokens for article and summary are 30 and 150. After filtering, we split the data into training, evaluation and testing data with the ratio of 8:1:1. Statistics of the dataset are shown in Table 1

The model is evaluated using the ROUGE package. We report rouge 1, rouge 2 and rouge L score as most papers do.

5.1. Training and inference

We use four layers in our stacked RNN and set 256 for all unit size including embedding size. All parameters are initialized

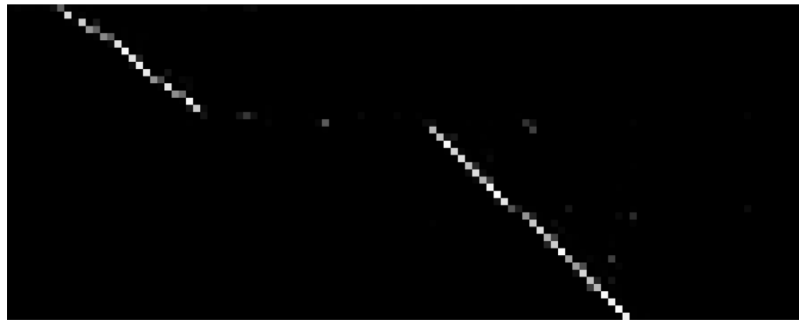


Fig. 5. Alignment history example, it shows that areas that have focus follow a descending pattern. White represents areas that are attended to, horizontal represents input sequence and vertical represents decoded sequence.

Table 3

An example of monotonic summarization. When using a stacked RNN, the output suffered from repetition. But when progressive energy and distance energy was added, there was no more repetition and also the content comes in the same order as the original document even when training data sometimes is not.

Original Text: cairo cnn at least 12 people were killed sunday, and more injured, in separate attacks on a police station, a checkpoint and along a highway in egypt's northern sinai, authorities said. six people, including one civilian, were killed when a car bomb exploded near the police station in al-arish, capital of north sinai, health ministry spokesman hossam abdel-ghafar told ahram online. he said 40 people were injured. ansar beit al-maqdis, an isis affiliate, claimed responsibility for the attack, which came hours after another operation that the group also claimed. in that earlier attack, a first lieutenant, a sergeant and four conscripts were killed when their armored vehicle was attacked on the highway from al-arish to sheikh zuweid in northern sinai, the military said. two other soldiers were injured and taken to a military hospital.

Reference: six people, including one civilian, are killed when a car bomb explodes near a police station. six others are killed when their armored vehicle is attacked on a highway in northern sinai. ansar beit al-maqdis, an isis affiliate, claim s responsibility.

StackedRNN: ansar beit al-maqdis, an isis affiliate, claimed responsibility for the attack. the attack came hours after another operation that the group also claimed. the attack came hours after another operation that the group also claimed.

StackedRNN + pre + dis: six people, including one civilian, were killed when a car bomb exploded near the police station in al-arish, capital of north sinai. ansar beit al-maqdis, an isis affiliate, claimed responsibility for the attack, which came hours after another operation that the group also claimed.

using the normal distribution. The model is trained using ADAM (Adaptive Moment Estimation) optimizer with a learning rate of 0.001. We train for 500000 steps with a batch size of 32, that is around 70 epochs. Using a Titan Xp GPU, it takes around 40 h to train the full model, which is less the that of pointer generator. The input document is truncated to a maximum size of 400. We set σ in Eq. (6) to 30, which mean that aligning weights 30 words away lose 40 percent of its value.

During our experiment, we find that hard alignment leads to better results. We approximate the hard alignment by normalizing the score before softmax activation in Eq. (6). Max-min normalization is used. This is only used during inference. Initial alignment is set to one hot vector with 1 at the beginning.

The model is based on the GNMT [24] model can be trained in parallel, we retain this ability so training speed has a possibility of being higher. Early stopping was not implemented and from the training loss (Fig. 6), we can see that the model has converged earlier than anticipated.

5.2. Results

Results on CNN dataset: Our results are shown in Table 2. From the results, we can see our model outperforms previous models that use monotonic attention [23] in both ROUGE 1 and ROUGE 2 score (only rouge 1 and 2 is reported). During our monotonic alignment summary experiment, the model without distance energy sometimes attends to areas in the end or middle at the very beginning thus discarding large potions of the document. Given that the initial alignment is one hot with 1 at the beginning, the model can generate sentences that are close to the start of the document (shown in Table 3).

We were able to produce results that were comparable to the pointer generator. Compared to the pointer generator, our

Table 4

Results on the Reddit dataset.

Model	Rouge-1	Rouge-2	Rouge-L
Stacked RNN	18.3	8.6	17.6
Hard monotonic attention [17]	19.6	10.3	18.9
Proposed monotonic	20.4	10.8	20.3

contexts were in order which is difficult to quantify and we rely on human observation for this. For this, we observe aligning distribution over history in some samples, we find that all attentions follow a descending pattern, as shown in Fig. 5. We observe over 100 alignments and find the same result.

We conduct the experiments on the model when using softmax, softmax + progressive energy and softmax + progressive energy + distance energy. We find that the last model did not skip large portions of context and produced summaries with a high rouge score.

Results on Reddits dataset: We conduct the experiment on the dataset with three models, The original stacked RNN model without any monotonic attention, Raffel [17] style monotonic attention and our proposed monotonic attention. For a fair comparison of each model, only part of the attention mechanism is changed with all other setting remaining the same. Two stacked RNN layers are used hidden size set to 128 throughout the network. We report Rouge 1, 2, and L score on the different models for performance comparison. Results are shown in the following Table 4. From the results, we can see that by using monotonic attention, the performance of summarization increases and our proposed method has the best performance. From our observation of the results of these models, Stacked RNN suffers from the repetition of generated sequences. Monotonic attention, on the other hand, does not have these problems. But in Raffel style

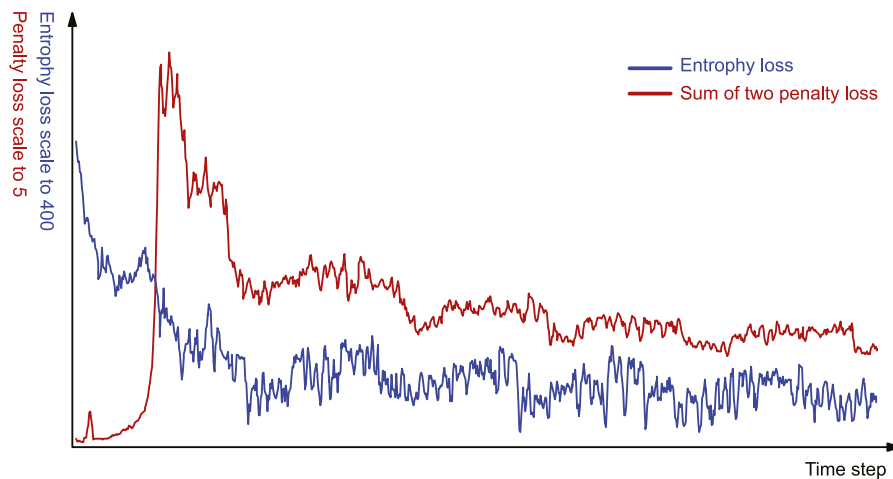


Fig. 6. Entropy loss value and penalizing loss value. Graph shows entropy loss decreases gradually, effects of penalizing loss is delayed.

attention, we find that decoding exits after a very small number of the token, this observation is due to the fact that monotonic attention does not attend to content to the left of the previous attention token. But problems arise when the observed token is very close to the end.

Our distance energy is effective in preventing the early ending of the summary. Though we are able to generate some summary, we find that the dataset is difficult to train a summarization model. Some the TL;DR samples are not related to the article, which some expressing feelings and emotions of the article.

5.3. Penalty loss of two energy

Our model is composed of two loss, the cross entropy based loss and penalizing loss. We find that the entropy loss decreased from the beginning, but the penalty loss had a rather unusual curve. At first, it has a small loss, then at some point, it jumps to a high value, from our observation, it is because at first, aligning distribution is rather uniform, according to our equations, the two energy loss will have very low values. But at some point when alignment starts being learned, the energy term comes into effect.

6. Conclusion

In this paper, we present a stacked RNN model using monotonic alignment for summarization. The model not only generates summaries that are in the same order as in the original document but can also be an alternate solution for repetition problem. The sub-word method is used to reduce the rate of UNK. To obtain the monotonic alignment, two energy functions were introduced to the scoring process during attention calculation and the corresponding penalty form was derived. We experiment on the CNN/Daily and Reddit dataset. The results show that we outperform the previous methods with monotonic alignment and are comparable to state-of-the-art results.

In future works, we would expand our work to include sentence level extractive methods to select highly informative parts of the original documents. Another interesting area is to use reinforcement learning to directly maximize the ROUGE score.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFB1401903). It also got partial support from National Engineering Laboratory for Cyber-learning and Intelligent Technology (China), and Beijing Key Lab

of Networked Multimedia, China. This work is supported by China National High-Tech Project (863) under grant (No. 2015AA015401). Beijing Key Lab of Networked Multimedia (China) also supports our research work. The work is supported by 973 Program, China (No. 2014CB340504), the State Key Program of National Natural Science of China (No. 61533018), NSFC-ANR, China (No. 61261130588), National Natural Science Foundation of China (No. 61402220, No. 61502221), the Philosophy and Social Science Foundation of Hunan Province, China (No. 16YBA323), Tsinghua University Initiative Scientific Research Program, China (No. 20131089256), Science and Technology Support Program, China (No. 2014BAK04B00), China Postdoctoral Science Foundation (No. 2014M550733), National Natural Science Foundation of China (No. 1309007), THU-NUS NExT Co-Lab, China and the Scientific Research Fund of Hunan Provincial Education Department, China (No. 19A439, No. 18B279, No. 16C1378).

References

- [1] A. See, P.J. Liu, C.D. Manning, Get to the point: summarization with pointer-generator networks, in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1073–1083.
- [2] R. Nallapati, F. Zhai, B. Zhou, SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4–9, 2017, San Francisco, California, USA., 2017, pp. 3075–3081.
- [3] R. Paulas, C. Xiong, R. Socher, A deep reinforced model for abstractive summarization, in: *International Conference on Learning Representations*, 2018, pp. 1–13.
- [4] Y.-C. Chen, M. Bansal, Fast abstractive summarization with reinforce-selected sentence rewriting, in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2018, pp. 675–686.
- [5] A.M. Rush, S. Chopra, J. Weston, A neural attention model for abstractive sentence summarization, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*, 2015, pp. 379–389, URL: <http://aclweb.org/anthology/D/D15/D15-1044.pdf>.
- [6] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, B. Xiang, Abstractive text summarization using sequence-to-sequence RNNs and beyond, in: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2016, pp. 280–290, <http://dx.doi.org/10.18653/v1/K16-1028>, URL: <http://www.aclweb.org/anthology/K16-1028>.
- [7] W.-T. Hsu, C.-K. Lin, M.-Y. Lee, K. Min, J. Tang, M. Sun, A unified model for extractive and abstractive summarization using inconsistency loss, in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, in: Long Papers, vol. 1*, 2018, pp. 132–141.

- [8] A. Cohan, F. Dernoncourt, D.S. Kim, T. Bui, S. Kim, W. Chang, N. Goharian, A discourse-aware attention model for abstractive summarization of long documents, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, 2018, pp. 615–621, URL: <http://aclweb.org/anthology/N18-2097>.
- [9] Y. Wu, B. Hu, Learning to extract coherent summary via deep reinforcement learning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 5602–5609.
- [10] A. Celikyilmaz, A. Bosselut, X. He, Y. Choi, Deep communicating agents for abstractive summarization, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, 2018, pp. 1662–1675, URL: <http://aclweb.org/anthology/N18-1150>.
- [11] P. Li, W. Lam, L. Bing, Z. Wang, Deep recurrent generative decoder for abstractive text summarization, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2017, pp. 2091–2100, URL: <http://aclweb.org/anthology/D17-1222>.
- [12] P. Li, Z. Wang, W. Lam, Z. Ren, L. Bing, Saliency estimation via variational auto-encoders for multi-document Summarization, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., 2017, pp. 3497–3503.
- [13] Y. Liu, C. Ouyang, J. Li, Ensemble method to joint inference for knowledge extraction, *Expert Syst. Appl.* 83 (2017) 114–121.
- [14] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, Arnetminer: extraction and mining of academic social networks, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 990–998.
- [15] T. Chung, B. Xu, Y. Liu, C. Ouyang, S. Li, L. Luo, Empirical study on character level neural network classifier for chinese text, *Eng. Appl. Artif. Intell.* 80 (2019) 1–7.
- [16] H. Wan, Y. Zhang, J. Zhang, J. Tang, Aminer: Search and mining of academic social networks, *Data Intell.* 1 (1) (2019) 58–76.
- [17] H. Jin, C. Li, J. Zhang, L. Hou, J. Li, P. Zhang, XLORE2: large-scale cross-lingual knowledge graph construction and application, *Data Intell.* 1 (1) (2019) 77–98.
- [18] Z. Cao, F. Wei, W. Li, S. Li, Faithful to the original: Fact aware neural abstractive summarization, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 4784–4791, URL: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16121>.
- [19] C. Li, W. Xu, S. Li, S. Gao, Guiding generation for abstractive text summarization based on key information guide network, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, 2018, pp. 55–60, URL: <http://aclweb.org/anthology/N18-2009>.
- [20] J. Lin, X. S.U.N., S. Ma, Q. Su, Global encoding for abstractive summarization, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, 2018, pp. 163–169, URL: <http://aclweb.org/anthology/P18-2027>.
- [21] A. Fan, D. Grangier, M. Auli, Controllable abstractive summarization, in: Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, Association for Computational Linguistics, 2018, pp. 45–54, URL: <http://aclweb.org/anthology/W18-2706>.
- [22] C. Raffel, M. Luong, P.J. Liu, R.J. Weiss, D. Eck, Online and Linear-Time Attention by Enforcing Monotonic Alignments, in: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, 2017, pp. 2837–2846.
- [23] C.-C. Chiu, C. Raffel, Monotonic chunkwise attention, in: International Conference on Learning Representations, 2018, pp. 1–16, URL: <https://openreview.net/forum?id=Hko85plCW>.
- [24] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's neural machine translation system: Bridging the gap between human and machine translation, *CoRR abs/1609.08144* (2016).
- [25] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473).
- [26] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, in: *ACL (1)*, The Association for Computer Linguistics, 2016, pp. 1715–1725.
- [27] M. Völske, M. Potthast, S. Syed, B. Stein, Tl; dr: Mining reddit to learn automatic summarization, in: Proceedings of the Workshop on New Frontiers in Summarization, 2017, pp. 59–63.