Ayana, Shen SQ, Lin YK *et al.* Recent advances on neural headline generation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(4): 768–784 July 2017. DOI 10.1007/s11390-017-1758-3

Recent Advances on Neural Headline Generation

Ayana [1,2,3,4], *Student Member, CCF*, Shi-Qi Shen [1,2,3], *Student Member, CCF*
Yan-Kai Lin [1,2,3,5], *Student Member, CCF*, Cun-Chao Tu [1,2,3,5], *Student Member, CCF*, Yu Zhao [1,2,3]
Zhi-Yuan Liu[1,2,3,5,*], *Senior Member, CCF*, and Mao-Song Sun[1,2,3,5], *Senior Member, CCF*

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

[2] *State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China*

[3] *Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China*

[4] *Department of Computer Information Management, Inner Mongolia University of Finance and Economics Hohhot 010000, China*

[5] *Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou 221009, China*

E-mail: ayn13@mails.tsinghua.edu.cn; {vicapple22, mrlyk423, tucunchao, zhaoyu62188}@gmail.com
liuzy@tsinghua.edu.cn; sms@mail.tsinghua.edu.cn

Received December 20, 2016; revised May 18, 2017.

Abstract Recently, neural models have been proposed for headline generation by learning to map documents to headlines with recurrent neural network. In this work, we give a detailed introduction and comparison of existing work and recent improvements in neural headline generation, with particular attention on how encoders, decoders and neural model training strategies alter the overall performance of the headline generation system. Furthermore, we perform quantitative analysis of most existing neural headline generation systems and summarize several key factors that impact the performance of headline generation systems. Meanwhile, we carry on detailed error analysis to typical neural headline generation systems in order to gain more comprehension. Our results and conclusions are hoped to benefit future research studies.

Keywords neural network, headline generation, data analysis

1 Introduction

Automatic text summarization is the process of creating a coherent, informative and brief summary of a document. Text summarization is expected to understand the central theme of the document and then output a condensed summary which contains as many key points of the original document as it can under a length limit. Text summarization approaches could be classified into two standard categories: extractive and generative. Most extractive summarization systems usually select a subset of actual sentences from the original documents as a summary. This leads to inherent drawbacks of extractive summarization, e.g., unable to generate coherent and compact summary in arbitrary length or shorter than one sentence. In contrast, generative summarization builds the semantic representation of a document and creates a summary with sentences which are not necessarily presented in the original document. Generative summarization needs to understand accurately and represent the semantics of the original document, and then generate informative summary according to document representation. Most previous studies heavily rely on modeling latent linguistic structures of input documents, via syntactic or semantic parsing, which always brings certain errors and degrades summarization quality.

Headline generation is within the area of automatic generation of summary, and its focus is the construction of headline-style abstracts from a single news ar-

Survey
Special Issue on Deep Learning
This work is supported by the National Basic Research 973 Program of China under Grant No. 2014CB340501, the National Natural Science Foundation of China under Grant Nos. 61572273, 61532010, and Microsoft Research Asia under Grant No. FY17-RES-THEME-017.

*Corresponding Author

ticle. It produces informative content describing the salient theme or event of the news article[1]. Recently deep learning has evolved into one of the most exciting and promising technologies in the field of artificial intelligence (AI), and brought great success in many natural language processing (NLP) tasks including headline generation[2-7]. Neural headline generation adopts an end-to-end encoder-decoder framework to model the entire headline generation process. The encoder reads and encodes a source article into a sequence of latent vectors (or a single vector). The decoder then outputs a summary word by word leveraging the information from the latent vectors.

In this work, we give an in-depth review of recent work on neural headline generation. The rest of our paper is structured as follows. In Section 2 we introduce standard components of a neural headline generation system, including the input representation methods, options of encoder and decoder, and existing training strategies. Further, we introduce the exploration of neural headline generation in different aspects including limited vocabulary size, length control, model architecture and so on. Section 3 introduces the widely used English and Chinese neural headline generation datasets in detail. In Section 4, we present a quantitative analysis of recent neural headline generation systems and explore the effect of different factors. In Section 5, we carry out manual analysis on development dataset to acquire more insights of neural headline generation systems. Besides, we also perform error analysis of different systems to explore the remaining problems for neural headline generation, which is expected to benefit the future research. Section 6 introduces related work of headline generation. Finally, this paper is concluded in Section 7.

2 Neural Headline Generation Model

Given an input document $\boldsymbol{X} = (x_1, \cdots, x_M)$, where each word x_i comes from a fixed vocabulary V, the neural headline generation model aims to take $\boldsymbol{X}$ as input, and generates a short headline $\boldsymbol{Y} = (y_1, \cdots, y_N)$ with length $N < M$ word by word. The log conditional probability can be formalized as:

$$\log \Pr(\boldsymbol{Y}|\boldsymbol{X}; \theta) = \sum_{j=1}^{N} \log \Pr(\boldsymbol{y}_j|\boldsymbol{X}, \boldsymbol{y}_{<j}; \theta),$$

where $\boldsymbol{y}_{<j} = (y_1, \ldots, y_{j-1})$ and θ indicates model parameters. That is, the j-th word y_j in the headline is generated according to all $\boldsymbol{y}_{<j}$ generated in past and the input document $\boldsymbol{X}$.

In the following, we will introduce the basic form for neural headline generation. The encoder-decoder architectures are presented in Subsection 2.1 and the training algorithms are introduced in Subsection 2.2. Furthermore, we explore some complicated problems for neural headline generation in Subsection 2.3.

2.1 Encoder-Decoder Models

A primary form of neural headline generation consists of three components: 1) an input representer which gives a representation of the input words, 2) an encoder which computes either a single vector or a sequence of vectors representing the original document, and 3) a decoder which generates one target summary word at a time.

2.1.1 Input Representation

First, a neural headline generation system projects discrete source article words into continuous vector space, and obtains the input representation $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_M\}$ of the source article:

$$\boldsymbol{E} = emb(1(\boldsymbol{X})), \tag{1}$$

where $1(\cdot)$ is a function to obtain the one-hot representation of word, and $emb(\cdot)$ represents the projection function to obtain word representation.

Word embeddings are low-dimensional real-valued vectors, which are not only capacity saving, but also able to reflect syntactic and semantic relationships between words. In this case, (1) could be re-written as:

$$\boldsymbol{E} = \boldsymbol{W}_x 1(\boldsymbol{X}),$$

where $\boldsymbol{W}_x \in \mathbb{R}^{D \times |V|_x}$ is a source word embedding matrix, D is the word embedding dimension size, and $|V|_x$ is the source-side vocabulary size.

Although word embedding is the most common way to represent input words, there are also efforts made to incorporate more complicated information of the input articles. For instance, word position is considered in [2], in addition to the word embedding. Specifically, the full embedding for each word is calculated as the sum of the word embedding and the word position embedding. Abstract meaning representation is utilized in [4] to incorporate syntactic and semantic information of input sentence into the headline generation model. More linguistic features are also considered in addition

to the word embedding to enrich linguistic features of source articles in [3]. The linguistic features include parts-of-speech tags, named-entity tags, and TF and IDF statistics of the words by utilizing additional embedding matrices to embed these features. We mainly focus on the comparison of different model architectures and training strategies; hence we only use word embedding to represent input words.

2.1.2 Encoder

The encoder encodes input representations into either a single vector or a sequence of vectors $\boldsymbol{H}$, which is also called source hidden states:

$$\boldsymbol{H} = enc(\boldsymbol{E}),$$

where $enc(\cdot)$ represents the encoder. We introduce different encoders in the following.

Bag-of-Words Encoder[7]. It is the most basic and simple model to embed an article into a fix-sized vector. It averages word embedding into a single vector, ignoring word order or relationships between words in a document. This encoder is simple yet unable to represent continuous phrases:

$$\boldsymbol{H} = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{E},$$

where $\boldsymbol{p} = [\frac{1}{M}]^M$ is a uniform distribution over the input words.

Convolutional Encoder[7]. It is proposed to embed input document using convolutional neural network. Specifically, [7] utilizes a time-delay neural network (TDNN) architecture, in which the final vector representing the original article is obtained by alternating between convolutional layer and pooling layer:

$$\boldsymbol{H} = tdnn(\boldsymbol{E}),$$

where $tdnn(\cdot)$ denotes the TDNN architecture. However, the convolutional encoder still has trouble to capture long-term dependencies. It is contrary to the nature of natural language by ignoring the connection between words in a sequence.

RNN Encoder. It is proposed to use recurrent neural network (RNN) to better model sequential information. RNN calculates the hidden state for each element of a sequence depending on its previous output. Hence, RNN can capture information about what has happened so far:

$$\begin{aligned}\boldsymbol{h}_i &= \phi(\boldsymbol{h}_{i-1}, \boldsymbol{e}_i)\\ &= \psi(\boldsymbol{W}_h\boldsymbol{h}_{i-1} + \boldsymbol{W}_x\boldsymbol{e}_i),\end{aligned} \tag{2}$$

where $\boldsymbol{e}_i$ is the word representation of the i-th input word, $\boldsymbol{h}_i$ denotes the i-th hidden state, $\phi(\cdot)$ represents the function to calculate the current hidden state, $\boldsymbol{W}_h$ and $\boldsymbol{W}_x$ are weight matrices, and $\psi(\cdot)$ is usually a nonlinear function. We omit the bias term for simplicity.

Although theoretically, RNN can make use of information in arbitrarily long sequences, it suffers from the exploding and vanishing gradient problems[8]. Recent studies on neural headline generation mostly utilize variants of plain RNN as the encoder.

In the case of GRU-RNN[9], (2) could be written as:

$$\begin{aligned}\boldsymbol{r}_i &= \sigma(\boldsymbol{U}_r\boldsymbol{h}_{i-1} + \boldsymbol{W}_r\boldsymbol{e}_i),\\ \boldsymbol{z}_i &= \sigma(\boldsymbol{U}_z\boldsymbol{h}_{i-1} + \boldsymbol{W}_z\boldsymbol{e}_i),\\ \tilde{\boldsymbol{h}}_i &= \tanh(\boldsymbol{U}_t(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1}) + \boldsymbol{W}_t\boldsymbol{e}_i),\\ \boldsymbol{h}_i &= (1-\boldsymbol{z}_i)\boldsymbol{h}_{i-1} + \boldsymbol{z}_i\tilde{\boldsymbol{h}}_i,\end{aligned}$$

where $\boldsymbol{z}_i$ is the update gate, $\boldsymbol{r}_i$ is the reset gate, $\tilde{\boldsymbol{h}}_i$ is the candidate activation, and $\odot$ is an element-wise multiplication. $\boldsymbol{U}_z$, $\boldsymbol{W}_z$, $\boldsymbol{U}_t$, $\boldsymbol{W}_t$, $\boldsymbol{U}_r$ and $\boldsymbol{W}_r$ are weight matrices. GRU-RNN is adopted in some headline generation systems[3,5-6].

In the case of LSTM-RNN[10], (2) could be written as:

$$\begin{aligned}\boldsymbol{h}_i &= \phi(\boldsymbol{h}_{i-1}, \boldsymbol{e}_i)\\ &= \boldsymbol{o}_i \tanh(\boldsymbol{C}_i),\\ \boldsymbol{o}_i &= \sigma(\boldsymbol{U}_o\boldsymbol{h}_{i-1} + \boldsymbol{W}_o\boldsymbol{e}_i),\\ \boldsymbol{f}_i &= \sigma(\boldsymbol{U}_f\boldsymbol{h}_{i-1} + \boldsymbol{W}_f\boldsymbol{e}_i),\\ \boldsymbol{i}_i &= \sigma(\boldsymbol{U}_i\boldsymbol{h}_{i-1} + \boldsymbol{W}_i\boldsymbol{e}_i),\\ \tilde{\boldsymbol{C}}_i &= \tanh(\boldsymbol{U}_c\boldsymbol{h}_{i-1} + \boldsymbol{W}_c\boldsymbol{e}_i),\\ \boldsymbol{C}_i &= \boldsymbol{f}_i\boldsymbol{C}_{i-1} + \boldsymbol{i}_i\tilde{\boldsymbol{C}}_i,\end{aligned}$$

where $\boldsymbol{o}_i$ is the output gate, $\boldsymbol{C}_i$ is the memory cell, $\boldsymbol{f}_i$ is the forget gate, $\boldsymbol{i}_i$ is the input gate, $\tilde{\boldsymbol{C}}_i$ is the memory content. $\boldsymbol{U}_o$, $\boldsymbol{W}_o$, $\boldsymbol{U}_c$, $\boldsymbol{W}_c$, $\boldsymbol{U}_f$, $\boldsymbol{W}_f$, $\boldsymbol{U}_i$ and $\boldsymbol{W}_i$ are weight matrices. Studies about headline generation systems adopting LSTM-RNN include [11-12].

Bidirectional RNN Encoder[3,6,12]. Conventional RNNs typically deal with text sequence from start to end and build the hidden state of each word only considering its previous words. It has been verified that the hidden state should also consider its following words as well. Hence, bidirectional RNN (BRNN)[13] is adopted to learn hidden states using both preceding and following words.

BRNN processes the input document in both forward direction and backward direction with two separate RNNs and obtains the forward hidden states $\overrightarrow{\boldsymbol{H}}$,

the backward hidden states $\overleftarrow{\boldsymbol{H}}$, and the final $\boldsymbol{H} = \overrightarrow{\boldsymbol{H}} \oplus \overleftarrow{\boldsymbol{H}}$ in which operator $\oplus$ indicates concatenation.

In this work, we implement four kinds of encoders, i.e., GRU-RNN, LSTM-RNN, GRU-BRNN and LSTM-BRNN encoders, to make a detailed comparison between different types of encoder.

2.1.3 Decoder

The decoder generates a headline word-by-word leveraging source hidden states $\boldsymbol{H}$:

$$\boldsymbol{Y} = dec(\boldsymbol{H}),$$

where $dec(\cdot)$ represents the decoder. At the j-th step of generation, i.e., generating the j-th headline word, the decoder updates its internal hidden state first and then computes the conditional distribution over the next target word. These operations are often based on source hidden states, previous decoder hidden states and previous output words. The decoder can be directly implemented as a neural network language model, or as various variants of recurrent neural network, as we will introduce below.

Neural Network Language Model[14]. It is adopted in [7] as the decoder to estimate the contextual probability of the next word:

$$\Pr(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{N} \Pr(\boldsymbol{y}_i|\boldsymbol{y}_c, \boldsymbol{H}),$$

where $\boldsymbol{y}_c$ is past c output words before $\boldsymbol{y}_i$, and we have

$$\begin{aligned} &\Pr(\boldsymbol{y}_i|\boldsymbol{y}_c, \boldsymbol{H}) \propto \exp(\boldsymbol{V}_{\text{nnlm}}\boldsymbol{s} + \boldsymbol{W}_{\text{nnlm}}\boldsymbol{H}), \\ &\boldsymbol{s} = \tanh(\boldsymbol{U}_{\text{nnlm}}\tilde{\boldsymbol{y}}_c), \end{aligned}$$

where $\tilde{\boldsymbol{y}}_c$ is concatenation of their word embeddings, $\boldsymbol{V}_{\text{nnlm}}$, $\boldsymbol{W}_{\text{nnlm}}$ and $\boldsymbol{U}_{\text{nnlm}}$ are weight matrices and $\boldsymbol{s}$ is the current hidden state. However, neural network language model decoder cannot consider the history information.

RNN Decoder. It decodes with recurrent neural network for better capturing sequential information. The RNN decoder can be formalized as:

$$\begin{aligned} \Pr(\boldsymbol{Y}|\boldsymbol{X}) &= \prod_{j=1}^{N} \Pr(\boldsymbol{y}_j|\boldsymbol{y}_{<j}, \boldsymbol{s}_j, \boldsymbol{H}) \\ &= \prod_{j=1}^{N} g(\boldsymbol{s}_j, \boldsymbol{y}_{j-1}, \boldsymbol{H}), \end{aligned}$$

where $g(\cdot)$ is a function that outputs the probability of $\boldsymbol{y}_j$ and $\boldsymbol{s}_j$ is the decoder hidden state which can be calculated as:

$$\boldsymbol{s}_j = f(\boldsymbol{s}_{j-1}, \boldsymbol{y}_{j-1}), \tag{3}$$

where $f(\cdot)$ is a function that calculates the current hidden state with regard to previous output word and hidden state.

Attention-Based RNN Decoder[2]. When generating a headline word, different input words usually make different contributions. To emulate the procedure, [15] proposes the attention mechanism in neural machine translation, which is later introduced in neural headline generation as well[2]. With the attention mechanism, (3) becomes:

$$\boldsymbol{s}_j = f(\boldsymbol{s}_{j-1}, \boldsymbol{y}_{j-1}, \boldsymbol{c}_j),$$

where $\boldsymbol{c}_j$ is the context vector, computed as a weighted average of $\boldsymbol{H} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_N)$:

$$\begin{aligned} \boldsymbol{c}_j &= \sum_{i=1}^{N} \alpha_{ji}\boldsymbol{h}_i, \\ \alpha_{ji} &= \frac{\exp(e_{ji})}{\sum_{k=1}^{N} \exp(e_{jk})}, \\ e_{ji} &= \rho(\boldsymbol{s}_{j-1}, \boldsymbol{h}_i), \end{aligned}$$

where α_{ji} is the weight assigned to the i-th source hidden state when decoding the j-th output word, e_{ji} is used to score how well $\boldsymbol{h}_i$ matches with $\boldsymbol{s}_{j-1}$, and $\rho(\cdot)$ is the scoring function.

Similar to the encoder, RNN variants, such as GRU-RNN and LSTM-RNN, are preferred instead of the plain RNN for the decoder. In our work, we implement attention-based GRU-RNN and LSTM-RNN decoders to examine the effect of different decoders.

2.2 Training Strategy

Existing neural headline generation models are mostly trained according to the maximum likelihood training strategy, which is essentially a word-level training strategy. There is also another line of sentence-level training strategy called minimum risk training[16], which can optimize model parameters with regard to evaluation matrices. We introduce the two strategies as follows.

2.2.1 Maximum Likelihood Estimation

In the traditional training strategy, the neural headline generation model parameters are optimized by maximizing the log likelihood of generated headlines over a set of training data D:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \{\mathcal{L}_{\text{MLE}}(\theta)\},$$

where

$$\mathcal{L}_{\mathrm{MLE}}(\theta) = \sum_{(\boldsymbol{X},\boldsymbol{Y})\in D} \log \Pr(\boldsymbol{Y}|\boldsymbol{X};\theta).$$

[16-17] point out that there are two shortcomings when utilizing MLE for encoder-decoder architecture. The first one is referred to as exposure bias[17]. That is, while the models are trained only on the training data distribution, an output word is predicted according to a correct previous word during training. However, at test time, an output word is predicted based on a predicted previous word which can be incorrect. Second, MLE usually uses the word-level cross-entropy loss to maximize the probability of the next word, which might hardly correlate well with the sentence level evaluation metric.

2.2.2 *Minimum Risk Training*

To tackle the problems of MLE training strategy, the minimum Bayes risk technique is introduced to tune neural model with respect to evaluation metrics. It aims at minimizing a sentence-wise loss function over the training data and has been widely used in many NLP tasks such as machine translation[16,18-20]. In our work, we implement the minimum risk training (MRT) to optimize the neural headline generation model as well.

Given a document $\boldsymbol{X}$, $\mathcal{Y}(\boldsymbol{X};\theta)$ is defined as the set of all possible headlines generated with parameter θ. Regarding $\boldsymbol{Y}'$ as the reference headline of $\boldsymbol{X}$, $\Delta(\boldsymbol{Y}',\boldsymbol{Y})$ represents the distance between $\boldsymbol{Y}$ and a generated headline $\boldsymbol{Y}'$. MRT defines the loss function as:

$$\mathcal{L}_{\mathrm{MRT}}(\theta) = \sum_{(\boldsymbol{X},\boldsymbol{Y})\in D} \mathbb{E}_{\mathcal{Y}(\boldsymbol{X};\theta)}\Delta(\boldsymbol{Y}',\boldsymbol{Y}).$$

Here $\mathbb{E}_{\mathcal{Y}(\boldsymbol{X};\theta)}$ indicates the expectation over all possible headlines. Thus the loss function of MRT can be further formalized as:

$$\mathcal{L}_{\mathrm{MRT}}(\theta) = \sum_{(\boldsymbol{X},\boldsymbol{Y})\in D} \sum_{\boldsymbol{Y}'\in\mathcal{Y}(\boldsymbol{X};\theta)} \Pr(\boldsymbol{Y}'|\boldsymbol{X};\theta)\Delta(\boldsymbol{Y}',\boldsymbol{Y}).$$

In this way, the training objective of MRT is to minimize the expected loss by perceiving the distance as a measure of assessing the overall risk:

$$\hat{\theta}_{\mathrm{MRT}} = \arg\min_{\theta} \{\mathcal{L}_{\mathrm{MRT}}(\theta)\}.$$

Nevertheless, it is usually time-consuming and inefficient to enumerate all possible instances. For simplicity, we draw a subset of samples $\mathcal{S}(\boldsymbol{X};\theta) \subset \mathcal{Y}(\boldsymbol{X};\theta)$ from the current probability distribution of generated headlines. Algorithm 1 shows how to sample from the current probability distribution of generated headlines. The loss function can be approximated as:

$$\begin{aligned}&\mathcal{L}_{\mathrm{MRT}}(\theta)\\ =&\sum_{(\boldsymbol{X},\boldsymbol{Y})\in D}\sum_{\boldsymbol{Y}'\in\mathcal{S}(\boldsymbol{X};\theta)} \times\\ &\frac{\Pr(\boldsymbol{Y}'|\boldsymbol{X};\theta)^{\epsilon}}{\sum_{\boldsymbol{Y}^{*}\in\mathcal{S}(\boldsymbol{X};\theta)}\Pr(\boldsymbol{Y}^{*}|\boldsymbol{X};\theta)^{\epsilon}}\Delta(\boldsymbol{Y}',\boldsymbol{Y}),\end{aligned} \tag{4}$$

where ϵ is a hyper-parameter that controls the smoothness of the objective function[18]. A proper ϵ value can significantly enhance the effectiveness of MRT[16]. Algorithm 2 shows how to update model parameter.

Algorithm 1. Sampling Process of MRT

Input: a training data pair $(\boldsymbol{X},\boldsymbol{Y})$ from training dataset D,
the limit on the size of sampling subset s,
the model parameter θ
Output: sampled subset $\mathcal{S}(\boldsymbol{X};\theta)$

$\mathcal{S}(\boldsymbol{X};\theta) \leftarrow \{\boldsymbol{Y}\}$;
$i \leftarrow 1$;
while $i \leqslant s$ **do**
  $\boldsymbol{y} \leftarrow \varnothing$; `// an empty candidate headline`
  $j \leftarrow 1$;
  while true **do**
    $y \sim \Pr(y_j|\boldsymbol{X},\boldsymbol{y}_{<j};\theta)$; `// sample the` j`-th word`
    $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \{y\}$;
    if $y = eos$ **then**
      break;`// stop generating if reach the end of sentence`
    end
    `// eos refers to end of sentence symbol`
    $j \leftarrow j + 1$;
  end
  $\mathcal{S}(\boldsymbol{X};\theta) \leftarrow \mathcal{S}(\boldsymbol{X};\theta) \cup \{\boldsymbol{y}\}$;
  $i \leftarrow i + 1$;
end

Algorithm 2. Training Process of MRT

Input: a set of training data D,
the limit on the size of sampling subset,
the set of model parameter θ',
the hyper-parameter ϵ that controls the smoothness of objective function
Output: optimized model parameter θ

Set initial value of parameter θ using the MLE parameter θ'
for $(\boldsymbol{X},\boldsymbol{Y}) \in D$ **do**
  Sample $\mathcal{S}(\boldsymbol{X};\theta)$;
  Compute gradient of $\sum_{\boldsymbol{Y}'\in\mathcal{S}(\boldsymbol{X};\theta)} \frac{\Pr(\boldsymbol{Y}'|\boldsymbol{X};\theta)^{\epsilon}}{\sum_{\boldsymbol{Y}^{*}\in\mathcal{S}(\boldsymbol{X};\theta)}\Pr(\boldsymbol{Y}^{*}|\boldsymbol{X};\theta)^{\epsilon}}\Delta(\boldsymbol{Y}',\boldsymbol{Y})$ w.r.t. θ;
  Update model parameter and get θ
end

MRT exploits the distance between two sentences to compose the loss function, which enables it to di-

rectly optimize model parameter concerning specific evaluation metric. Headline generation is fundamentally a sub-task of summarization. Therefore, it is reasonable to take ROUGE[21], the most widely used evaluation metric for document summarization, to compose the distance in MRT. The basic idea of ROUGE is to count the number of overlapping units between computer-generated summaries and the reference summaries, such as overlapped N-grams, word sequences, and word pairs. We measure the distance $\Delta(\boldsymbol{y}', \boldsymbol{y})$ with ROUGE in our work, and more details about ROUGE are introduced in Subsection 4.1.

2.3 Improvement for Neural Headline Generation

Many researchers have been focusing on other aspects to improve the performance of neural headline generation system as well. We will introduce them in details in this subsection.

2.3.1 Limited Vocabulary Size

At each generation step of the decoder, the output word is selected according to the probability distribution over the whole target side vocabulary, which is the most time and capacity consuming part of the system. To balance the performance and the efficiency, most systems would keep a fix-sized target vocabulary concerning word frequency. Infrequent words would be replaced by a unique token "UNK", meaning unknown words. While there are only a few unknown words in the headline, this approach would work well. However, it has been observed that the infrequent words are usually proper nouns or named-entities that have deterministic influence on the meaning of the headline. Mapping them into a unified "UNK" token regardless of other consideration would hurt system performance. In this subsection, we introduce studies that make an effort to deal with the limited vocabulary size problem in neural headline generation system.

[6] proposes a new model called COPYNET, which is an encoder-decoder architecture equipped with copying mechanism. The motivation behind this design is to simulate the human behavior that people tend to repeat named entities or even longer phrases when communicating, especially when they are unfamiliar with those named entities or phrases. In the canonical attention based encoder-decoder model, to output a word, the model would consider only a generating mode. COPYNET also considers an additional copy mode, so that the model could accommodate both generating and copying. [22] points out that unknown words are hurting the performance of most NLP applications, including statistical-based and neural network based applications. Thus [22] proposes to solve the unknown word related problem in end-to-end neural network, i.e., when predicting an output word, the model first makes a decision whether to pick a word from target side vocabulary or copy one from source input. [3] tries to deal with the rare keywords using the pointer network[23]. The models mentioned above[3,6,22] share the same essence: utilizing input words that are not in the target vocabulary to expand the limited target vocabulary size.

Besides, [3] explores another way to deal with the limited vocabulary size, inspired by [24]. Specifically, at each mini-batch, [3] restricts the decoder-vocabulary to words in the source documents of the batch. If the decoder-vocabulary size does not reach a fixed size, more words of the target dictionary are added concerning frequency. This technique is referred to as LVT (large vocabulary trick).

In our work, we explore an easier "UNK replace" method during testing, which is also inspired by [24]. That is, when the model generates a "UNK" token, we replace it based on alignment information.

Another way to deal with the computational complexity from very large vocabulary is to utilize a character-level model, as proposed in [5]. [5] treats source articles and target headlines as sequences of characters without any explicit word segmentation. The work of [5] is based on a Chinese dataset. Hence we also adopt character level model for Chinese systems.

2.3.2 Length Control

Neural headline generation aims at highly summarizing a longer article into a short and capable headline that includes the most salient information in the original article. Depending on different application scenarios, there is one important attribute that headlines should possess: the length of a headline can be managed within the desired range. It is intractable to keep as much salient information as possible and to make the summary length achieve a pre-defined size at the same time. To address this problem, [11] proposes four model variants. Two of them are based on different decoding procedures without model architecture modification. The other two are learning-based, i.e., the models take the desired length information as input and encode it into the model architecture. The best performing model, LenEmb, takes the length limit

embedding as additional model input. When decoding, LenEmb maintains a remaining length embedding, which is changing along with the decoding process, to control how many bytes of outputs will be generated.

2.3.3 Model Architecture Exploration

Researchers also explore if there are more appropriate model architectures for neural headline generation.

In the attention-based encoder-decoder model, the input sequence has to be read in and encoded before the model produces any output, which is time- and capacity-consuming. Inspired by the HMM word alignment model[25-26], [27] implements a new neural headline generation architecture to deal with the above-mentioned bottleneck. Specifically, when reading in an input signal, the model has to decide whether to read the following input or generate an output signal, depending on the transition probability. Note that the model is online sequence-to-sequence model only when the encoder is unidirectional.

[12] proposes a semi-supervised architecture for neural headline generation. Specifically, [12] combines a generative unsupervised model (auto-encoding sentence compression, ASC) that takes variational auto-encoding framework as inference algorithm, and a discriminative supervised model (forced attention sentence compression, FSC) together. As a result, the system can model language as a discrete latent variable in the variational auto-encoding framework, and the resultant generative model can exploit both supervised and unsupervised data in headline generation task.

2.4 Comparison with Neural Machine Translation

Among many tasks in the field of natural language processing, the most related task to headline generation is machine translation. Machine translation aims at translating an input sequence of words in the source language to an output sequence of words in the target language. Headline generation can also be treated as a special kind of translation process: translating an input sequence of words in a source document to an output sequence of words in a targeted headline. The source document and the target headline are in the same language though. Due to the similarity, researchers have explored the possibility of applying machine translation models to headline generation[28]. Also, the officially adopted headline generation evaluation metric ROUGE is also inspired by BLEU, a standard machine translation evaluation metric.

Neural network based encoder-decoder architecture is first proposed in machine translation and achieves state-of-the-art performance compared with traditional statistical models. According to the aforementioned similarity between machine translation and headline generation, it is a natural choice to adopt the encoder-decoder architecture for headline generation as well.

Despite the similarity, there are some important differences between machine translation and headline generation. In machine translation, the target output is expected to keep the original information of the source input to the greatest extent. Hence the lengths of the source input and the target output are close to each other. In headline generation, on the other hand, the target headline only remains the most salient parts of the source document. As a result, the headline is typically much shorter than the original document. Hence, although neural headline generation systems share a similar framework with neural machine translation systems, the significance is quite different, which also indicates the flexibility of neural models.

3 Corpora

In this section, we introduce the standardized English test set and the automatically composed training sets (for English and Chinese respectively) in details.

3.1 DUC Data

The headline generation task is standardized in Task-1 of DUC2003 and DUC2004①. The DUC2003 dataset consists of 624 news articles from Associated Press Wire services and New York Times. Following [7], the DUC2003 Task-1 data is often utilized as development set, to select hyper-parameters. The DUC2004 Task-1 data is usually taken as test set, and it consists of 500 news articles. Each article of these two datasets is paired with four human-generated reference headlines. Table 1 shows the statistics of them.

Table 1. Data Statistics of the DUC Datasets

Dataset	Statistics		
	art.num	art.avg.tok	head.avg.tok
DUC2003	624	35.37	10.03
DUC2004	500	35.56	10.43

Note: art.num, art.avg.tok and head.avg.tok refer to the number of articles, average token numbers in each article, and average token numbers in each headline respectively.

①http://duc.nist.gov/, May 2017.

3.2 English Gigaword

The English training dataset, which is introduced in [7], is made from English Gigaword[29]②. English Gigaword is one of the largest static corpora of English news, and it has the following layers of annotation: sentences segmentation and tokenization, treebank-style constituent parse trees annotation, syntactic dependency trees annotation, named entities annotation, and in-document coreference chains annotation. It consists of nearly 10 million news articles from seven news outlets, with a total of more than four billion words. To utilize it for training headline generation model, [7] carries out a preprocessing and filtering procedure. [7] also releases the corresponding preprocessing script to benefit others. Specifically, the processing steps include: 1) filtering out news articles that overlap with DUC2003 and DUC2004 datasets; 2) filtering out headlines including bylines, extraneous editing marks and question marks; 3) utilizing annotations for tokenization and sentence separation to get word tokens; 4) opting the first sentence of each news article and pairing it with its corresponding headline as an article-headline pair; 5) lower-casing, replacing all digit characters with #, and replacing word types seen less than five times with "UNK". The data statistics after these preprocessing steps is shown in Table 2. In our experiments, we also follow the aforementioned preprocessing steps.

3.3 LCSTS

LCSTS[5], the Large-Scale Chinese Short Text Summarization dataset, is commonly used for Chinese headline generation. It is constructed based on Sina Weibo③, a popular social medium in China. On Sina Weibo, some weibos are associated with headline-like contents encapsulated in parentheses, which is a natural fit to the data format of the headline generation task. Since Sina Weibo is open to any users, to make sure the crawled weibos are clean, formal and informative, only those weibos from 50 very popular certified organization users and their blue verified followers are collected. After filtering out too short weibos (less than 80 characters) and inappropriate headlines (out of range of 10∼30 characters), and manually annotating partial weibos, the dataset is categorized into three parts, the bottom row of Table 2 shows the statistics of the dataset.

4 Experiments

In this section, we introduce experiments that we conduct to help better understand neural headline generation systems.

4.1 Evaluation Metric

ROUGE[21], standing for Recall-Oriented Understudy for Gisting Evaluation, is the most widely used evaluation metric for document summarization. It is also adopted to evaluate headline generation system performance by Document Understanding Conference (DUC), a large-scale summarization evaluation sponsored by NIST (National Institute of Standards and Technology) of United States Department of Commerce. Inspired by the evaluation metric of machine translation, the basic idea of ROUGE is to count the number of overlapping units between computer-generated summaries and the reference summaries.

ROUGE-N is an N-gram recall between a generated summary $\boldsymbol{Y}'$ and the reference summary $\boldsymbol{Y}$, which can be formulated as:

$$\textit{ROUGE-N} = \frac{\sum_{\text{gram}_N \in \boldsymbol{Y}} C_M(\text{gram}_N)}{\sum_{\text{gram}_N \in \boldsymbol{Y}} C_{\boldsymbol{Y}}(\text{gram}_N)}, \tag{5}$$

where gram_N represents the N-gram, $C_{\boldsymbol{Y}}(\text{gram}_N)$ indicates the maximum number of co-occurring N-grams in $\boldsymbol{Y}'$, and $C_M(\text{gram}_N)$ denotes the number of N-grams in $\boldsymbol{Y}$. In the experiment, we take two types of N-gram into consideration, i.e., uni-gram and bi-gram, corresponding to ROUGE-1 and ROUGE-2 respectively.

Table 2. Data Statistics of English Gigaword and LCSTS Datasets

Language	Dataset	Statistics				
		Train	Valid	Test	art.avg.tok	head.avg.tok
English	English Gigaword	3 799 588	394 622	381 197	31.35	8.23
Chinese	LCSTS	2 400 591	10 666	1 106	103.68	17.86

Note: "Train", "Valid" and "Test" represent the number of article-healdine pairs in the training, validation and test set respectively. art.avg.tok refers to average token number of articles, head.avg.tok refers to average token number of headlines, and both these statistics are based on the training set.

②https://catalog.ldc.upenn.edu/LDC2012T21, May 2017.

③http://www.weibo.com/, May 2017.

ROUGE-L is a longest common sub-sequences based F-measure to compare the similarity between two sentences. It can be formalized as:

$$\begin{aligned} ROUGE\text{-}L &= \frac{(1+\beta^2) r_{\mathrm{L}} p_{\mathrm{L}}}{r_{\mathrm{L}} + \beta^2 p_{\mathrm{L}}}, \\ r_{\mathrm{L}} &= \frac{Lcs(\boldsymbol{Y}', \boldsymbol{Y})}{Len(\boldsymbol{Y})}, \\ p_{\mathrm{L}} &= \frac{Lcs(\boldsymbol{Y}', \boldsymbol{Y})}{Len(\boldsymbol{Y}')}, \end{aligned} \tag{6}$$

where $Lcs(\boldsymbol{Y}', \boldsymbol{Y})$ is the length of the longest common subsequence between $\boldsymbol{Y}'$ and $\boldsymbol{Y}$, $Len(\boldsymbol{Y})$ is the length of $\boldsymbol{Y}$ and β is the harmonic factor between recall r_{L} and precision p_{L}.

Following [2-3, 7], for DUC2003 and DUC2004, we report recall scores of ROUGE-1, ROUGE-2 and ROUGE-L with official 75 bytes ceiling limit. Meanwhile, following [2-3, 7], for Gigaword test set, we report full-length F-measure scores of ROUGE-1, ROUGE-2 and ROUGE-L. Since a shorter summary tends to get a lower recall score, when testing on DUC datasets, we set the minimum length of a generated headline to 10. Note that we report 75 bytes capped recall scores only. In this case, summaries longer than 75 bytes obtain no bonus on recall scores. Due to that the full-length F-measure makes the evaluation result unbiased to summary length, we set no limitation to the headline length when testing on Gigaword test set.

For Chinese, we report full-length F-measure scores following previous work[5-6]. We set no length limitation on Chinese experiments either.

4.2 Implementation Details

We conduct several experiments to compare different encoders, decoders and training strategies and the experimental setup is as follows. For English models, we set the word embedding dimension to 600, the hidden unit size to 1 000 and the vocabulary size to 30 000. The corresponding values for Chinese models are 400, 500 and 3 500 respectively. In MLE systems, model parameters are randomly initialized and then updated during training. In MRT systems, model parameters are initialized using the optimized parameters from the MLE systems. When training English MRT systems, we adopt negative recall value of ROUGE-1, ROUGE-2 and ROUGE-L to compose $\Delta(\boldsymbol{Y}', \boldsymbol{Y})$. For Chinese systems, we utilize negative F-measure value of ROUGE-1, ROUGE-2 and ROUGE-L to compose $\Delta(\boldsymbol{Y}', \boldsymbol{Y})$. In particular, the size of subset $\mathcal{S}(\boldsymbol{X}; \theta)$ in (4) has a great impact on the performance. When the size is too small, the sampling will be insufficient. When the size is too large, the learning time will grow correspondingly. In this paper, we set the size to 100 to achieve a trade-off between effectiveness and efficiency. These samples are drawn from the probability distribution of generated headlines by the up-to-date system④. We use AdaDelta[30] to adapt learning rates in stochastic gradient descent for both MLE and MRT systems. We utilize no dropout or regularization, but we take gradient clipping during the training and the training is early stopped based on the DUC2003 data. All models are trained on GeForce GTX TITAN GPU. For the MLE system on the English datasets, it takes about 2.5 hours for each 10 000 iterations. For the MRT system, it takes about 3.75 hours. During the testing, we use the beam search of size 10[2] to generate headlines.

4.3 Baseline Systems

4.3.1 English Systems

ABS and ABS+[7] both utilize weighted bag-of-words encoder and NNLM decoder. The difference is that ABS+ extracts additional N-gram features at the word level to revise the output of the ABS model.

Luong-NMT[31] exploits 2-layer attention based LSTM-RNN architecture.

words-lvt2k-1sent[3] is based on GRU-BRNN encoder and attention-based GRU-decoder, and also utilizes the large vocabulary trick.

ABS+AMR[4] utilizes the same model architecture as ABS and meanwhile takes input article parsing information in addition.

RAS-Elman and RAS-LSTM[2] both utilize convolutional encoders that take word position as extra input information. RAS-Elman is based on the Elman-RNN decoder, while RAS-LSTM is based on the LSTM-RNN decoder.

LenEmb[11] tries to solve the output length controlling problem.

ASC+FSC[12] is a semi-supervised neural headline generation system.

MLE and MRT denote the systems we implement. They are both composed of GRU-BRNN encoder and attention-based GRU-RNN decoder. The difference between them is the different training strategy.

④An alternative subset building strategy is to choose top-k headlines. Considering the efficiency and parallel architecture of GPUs, we opt sampling.

4.3.2 Chinese Systems

RNN-Context(W) and RNN-Context(C)[5] are word-based and character-based models respectively. The basic model architecture is the GRU-BRNN encoder and the attention-based GRU-RNN decoder.

COPYNET(W)[6] incorporates copying mechanism into a sequence-to-sequence framework, which replicates certain segments from the input sentence into the output sentence.

MLE(C) and MRT(C) remain the same meaning as in English systems. (C) means that the system is character-based.

4.4 Evaluation Results

In this subsection, we illustrate the overall evaluation results in both English and Chinese test sets. $R1_R$, $R2_R$ and RL_R stand for recall scores of ROUGE-1, ROUGE-2 and ROUGE-L respectively. $R1_{F1}$, $R2_{F1}$ and RL_{F1} stand for F-measure scores of ROUGE-1, ROUGE-2 and ROUGE-L respectively.

We discuss the performances in details to gain more insights about what works for neural headline generation systems. Since the architectures vary in different systems, we make the comparison between similar systems and find out what is working. When necessary, we also implement further auxiliary experiments to verify our observations.

4.4.1 Results on English Corpus

From Table 3 and Table 4, we observe the following key factors that affect the performance of headline generation systems.

Linguistic Feature. words-lvt2k-1sent[3] takes additional linguistic features such as POS tag, NER tag, TF-IDF statistics in their input representation. ABS+AMR[4] includes parsing information in their input representation. ABS+[2] is a system that adopts extractive tuning. Comparing words-lvt2k-1sent with MLE, and ABS+AMR with ABS+, we find that adding

Table 3. ROUGE Scores on DUC2004 and Gigaword English Test Sets

System	DUC2004			Gigaword		
	$R1_R$	$R2_R$	RL_R	$R1_{F1}$	$R2_{F1}$	RL_{F1}
ABS[7]	26.55	7.06	22.05	29.55	11.32	26.42
ABS+[7]	28.18	8.49	23.81	29.76	11.88	26.96
Luong-NMT[31]	28.55	8.79	24.43	33.10	14.45	30.71
words-lvt2k-1sent[3]	28.35	9.46	24.59	32.67	15.59	30.64
ABS+AMR[4]	28.80	7.83	23.62	-	-	-
RAS-LSTM[2]	27.41	7.69	23.06	32.55	14.70	30.03
RAS-Elman[2]	28.97	8.26	24.06	33.78	15.97	31.15
LenEmb[11]	26.73	8.40	23.88	-	-	-
ASC+FSC[12]	-	-	-	34.16	15.94	31.92
MLE	24.92	8.60	22.55	32.67	15.23	30.56
MRT	**30.41**	**10.87**	**26.79**	**36.54**	**16.59**	**33.44**

Table 4. Model Architectures Corresponding to Table 3

System	Input	Encoder	Decoder	Others
ABS[7]		Weighted bow	NNLM+Att	
ABS+[7]		Weighted bow	NNLM+Att	Extractive feature
Luong-NMT[31]		2-layer LSTM	2-layer LSTM+Att	
words-lvt2k-1sent[3]	+Linguistic features	GRU-BRNN	GRU+Att	LVT
ABS+AMR[4]	+Parsing information	Weighted bow	NNLM+Att	
RAS-LSTM[2]	+Position embedding	Convolutional	LSTM+Att	
RAS-Elman[2]	+Position embedding	Convolutional	Elman+Att	
LenEmb[11]	+Length information	LSTM-BRNN	LSTM+Att	
ASC+FSC[12]		LSTM-BRNN	LSTM+Att	
MLE		GRU-BRNN	GRU+Att	UNK replace
MRT		GRU-BRNN	GRU+Att	UNK replace

Note: In the "Input" column, a blank bar means a model only takes word embeddings as the encoder input. In the "Others" column, a blank bar means a model takes no other special techniques. Att denotes attention mechanism and weighted bow means weighted bag-of-words.

additional linguistic features of input could help in improving system performance. Also these features benefit the interpretability of neural headline generation system.

Encoder Choice. Those systems which utilize BRNN seem to have higher ROUGE-2 and ROUGE-L scores. The most obvious example is that on both DUC and Gigaword, the MLE system obtains higher ROUGE-2 and ROUGE-L scores as compared with the extractively tuned ABS+ system. Hence we believe that the bidirectional encoder has the ability to encode both forward and backward information as it claims, and consequently leads to more coherent target output. To verify this observation, we implement four encoders while fixing the decoder to be the same (attention-based GRU-RNN). The experimental results are shown in Table 5. We can find that BRNNs get higher $R2_R$ and $R\mathrm{L}_R$ scores, which are important indication of coherency. Hence the aforementioned observation is verified. For those encoders that are not based on recurrent neural network, the advantages might be the light weightiness as mentioned in [2]. That is, when obtaining comparable system performance, these systems require fewer parameters.

Table 5. Effect of Different Encoders on DUC2004

Encoder	Evaluation Metric		
	$R1_R$	$R2_R$	$R\mathrm{L}_R$
GRU-RNN	25.42	7.64	22.44
GRU-BRNN	24.92	**8.60**	22.55
LSTM-RNN	**26.14**	8.25	23.17
LSTM-BRNN	26.09	8.44	**23.18**

Decoder Choice. ABS, ABS+, and ABS+AMR take non-RNN architecture as their decoder. We find that the overall performance of them is inferior to that of those RNN-based ones. Hence when making a decision about the decoder, RNN-based decoders are recommended. [7] reports that their attention-based encoder performs much better than the bag-of-words and convolutional encoders. Combining the fact that the attention mechanism has achieved significant improvement in NMT, adopting the attention mechanism in a model is a smart choice. Furthermore, we carry out comparative experiments to explore the effectiveness of GRU decoder and LSTM decoder, and the results are described in Table 6. As we can see from the table, when exploiting the same encoder, the GRU-RNN decoder always gets better performance. Therefore, we believe that the GRU-RNN decoder is more suitable for headline generation than the LSTM-RNN decoder.

Table 6. Effect of Different Decoders on DUC2004

Encoder	Decoder	Evaluation Metric		
		$R1_R$	$R2_R$	$R\mathrm{L}_R$
GRU-BRNN	GRU	24.92	**8.60**	22.55
	LSTM	24.49	7.23	21.64
LSTM-BRNN	GRU	**26.09**	8.44	**23.17**
	LSTM	25.51	7.80	22.59

Training Strategy. Analyzing training strategy, we observe that the MRT system, the only system optimized with minimum risk training algorithm, significantly improves over the MLE system, and is also superior to other systems. This suggests that the sentence-level optimizing algorithm is much more efficient than the word-level optimizing algorithm. As described in Subsection 2.2.2, the distance $\Delta(\boldsymbol{Y}', \boldsymbol{Y})$ in the loss function of MRT is computed by the negative value of ROUGE. We also conduct experiments to investigate the effect of utilizing various distance measures in MRT. Table 7 shows the experimental results on DUC2004 using different evaluation metrics. We find that all MRT systems consistently outperform the MLE system, which indicates that the MRT technique is robust when loss function varies. We also find that when using $-R1_R$ and $-R\mathrm{L}_R$ as distance measures, the corresponding $R1_R$ and $R\mathrm{L}_R$ scores reach the highest. However when using $-R2_R$, the corresponding system does not get the best $R2_R$ score as we expected. Instead, the $-R\mathrm{L}_R$ system gets the best $R2_R$ score. One possible explanation is that $R\mathrm{L}_R$, which measures the longest common string between two sentences, also can measure bi-gram information.

Table 7. Effect of Using Different Distance Measures in MRT on DUC2004

System	Evaluation Metric		
	$R1_R$	$R2_R$	$R\mathrm{L}_R$
MLE	24.92	8.60	22.55
$-R1_R$	**29.84**	10.24	26.33
$-R2_R$	28.35	10.43	25.28
$-R\mathrm{L}_R$	29.80	**10.87**	**26.35**

Output Length[11]. It tries to solve a basic problem of headline generation under the neural network architecture, i.e., taking control over the output length. From Table 3, we find that the ROUGE scores of [11] are inferior to most systems. One possible explanation is that the test dataset (DUC2004) is not very suitable for the aim of this work, i.e., there is no specific category classified according to different length values in the test dataset.

Corpus. We also find that the improvement of different systems is not very consistent on the DUC and Gigaword datasets. For example, ABS+ shows significant improvement over ABS on the DUC datasets, but is not so significant on Gigaword. Since ABS+ is an enhanced version of ABS, i.e., ABS+ parameters are tuned according to the DUC2003 dataset, the superior performance on DUC seems to be reasonable. However, the same observation could be found when comparing words-lvt2k-1sent and MLE. We suspect that the reason might be relative to the reference headlines. The DUC datasets have four manual references for each article when Gigaword only has one. According to [21], more references lead to more stable evaluation. Hence we believe results on DUC datasets are more reliable.

4.4.2 Results on Chinese Corpus

We have illustrated most English headline generation experimental results so far and made comparison among different encoders, decoders and training strategies. However, there is an important problem in accomplishing Chinese neural headline generation, i.e., the segmentation problem. We conduct auxiliary experiments on Chinese to compare the character-based and the word-based model. Table 8 shows the evaluation results of headline generation on Chinese test sets⑤. The five systems listed in the table share the same basic model architecture, i.e., GRU-BRNN encoder + GRU-RNN decoder + attention mechanism. The differences include: 1) RNN context(W) and COPYNET(W) are word-based, and the others are character-based; 2) only COPYNET(W) incorporates the Copy mechanism; 3) only MRT(C) is trained with minimum risk training algorithm, and the others are trained with maximum likelihood estimation algorithm; 4) RNN context(C) and our MLE system differ in decoding method during test time.

Table 8. ROUGE Scores on Chinese Test Set

System	LCSTS		
	$R1_{F1}$	$R2_{F1}$	RL_{F1}
RNN context(W)[5]	26.8	16.1	24.1
RNN context(C)[5]	29.9	17.4	27.2
COPYNET(W)[6]	35.0	22.3	32.0
MLE(C)	34.9	23.3	32.7
MRT(C)	**38.2**	**25.2**	**35.4**

Note: (W) and (C) mean word-based and character-based respectively.

Due to the fact that the words are not delimited by white space in Chinese, word-segmentation is often a necessary pre-processing step. The error of word segmentation would be brought into a model inevitably. In addition, the segmentation results would influence the fix-sized vocabulary as well. One way to solve the problem is to directly separate an article character by character. In this way, there will be no word-segmentation error in the system, and the vocabulary could be down-sized substantially as well. Comparing RNN context(W) and RNN context(C), we find that the character-based model performs significantly better than the word-based model. We also find that, although COPYNET(W) is word-based, its ROUGE-2 and ROUGE-L scores are notably better than the RNN context(W), even better than the character-based RNN context(C). This indicates the effectiveness of Copy mechanism in handling word-level or phrase-level information. With the strength of the sentence-level optimization, the MRT system improves the ROUGE scores up to over 3 points compared with the MLE system and consistently outperforms other baseline systems. This again proves the effectiveness of minimum risk training algorithm in neural headline generation systems.

5 Analysis

The MRT training strategy could significantly improve the neural headline generation system performance, and our implemented MRT system achieves the state-of-the-art in existing neural headline generation systems. In order to further understand our MRT system, and benefit later research, we conduct an in-depth comparison between the MLE and MRT systems and try to answer the following questions. 1) What can the neural headline generation systems we implemented learn? 2) What kinds of errors are the neural headline generation systems making?

5.1 Data Analysis

By looking over the per-document performance of the systems, we find that the ROUGE scores change dramatically. Hence we decide to investigate about the source documents and their corresponding headlines, and try to discover whether there is a connection between data property and system performance. To study this, we randomly sample 100 examples from the development set of the LCSTS dataset for analysis.

⑤The MRT result reported here is obtained by taking the negative F-measure score of ROUGE-1 as the loss function. Several related experimental results are not given due to the length limit.

5.1.1 Decomposing the Examples

After carefully analyzing the 100 examples, we classify them into six categories. There might be cases that an example satisfies more than one category. In these cases, we classify them into the earliest one and Table 9 provides the percentage of each category. The categories are as follows.

Proper Subset. Headline words are all from the original article.

Expression Quotation. This category directly refers to the speech fragment of someone while describing an event, so that the description becomes more vivid. This narrative approach is usually accompanied with post positioned subject.

Paraphrasing. Although headline words are not exactly the same with the ones in the original article, they remain the same meanings obviously. Words in this kind of headline are usually reordered to make the headline coherent.

Mathematical Reasoning. The original article includes digital information; hence one should conduct simple mathematical operation, such as counting or adding, to get the corresponding digit in the headline.

Domain Knowledge. In this category, the original article writes about contents that are targeted at specific readers, financial report for instance. To manually generate a concrete headline for this category, one should be provided with certain domain knowledge.

Weak Clue. Headlines are supposed to highly summarize the original article, so that the readers can determine whether or not to continue reading this article. Hence, it is unavoidable that there are many cases in which headlines are intended to attract the public eye, but have no much relationship with the original articles. Basically target summaries of this category are treated as "unobtainable".

5.1.2 Per Category Performance

We report the evaluation results of two systems based on the above categorization in Table 9. From the table, we find the following facts. 1) The proper subset and expression quotation cases are quite simple and both systems get promising scores compared with other cases. 2) On the paraphrasing cases, the MRT system performs much better than the MLE system, which confirms that the MRT system is good at capturing the integrated information. 3) The mathematical and domain knowledge cases require basic reasoning ability and domain knowledge to generate a coherent headline. Hence, both systems get lower scores as compared with other cases. However, since these systems do not incorporate any domain knowledge, it is inappropriate to evaluate the system performance with regard to reference headline with domain knowledge. 4) For weak clue cases, since the golden summaries have not much relationship with the original articles, even for human, it is unable to generate the golden outputs.

To address the issue mentioned in 3) and 4), plus to carry out more reasonable, objective and fair analysis, we further build reference headlines manually for those domain knowledge and weak clue cases (62 examples in total).

5.2 Manually Reference

We request four annotators to generate headlines for the 62 examples manually. In fact, writing a headline is a very subjective task, and headlines generated by different annotators could be quite different. Hence, we gives the following rules to better control the subjective difference.

1) Given a weibo which includes less than 140 characters, annotators need to generate a headline within 10∼20 characters.

2) The headline should contain the most salient content of the original weibo.

Table 9. Per-Category Performance of MLE and MRT

No.	Category	Statistics (%)	MLE			MRT		
			$R1_{F1}$	$R2_{F1}$	RL_{F1}	$R1_{F1}$	$R2_{F1}$	RL_{F1}
1	Proper subset	10	63.5	51.1	60.4	67.0	52.8	64.4
2	Expression quotation	10	50.7	33.9	45.1	56.7	39.6	51.4
3	Paraphrasing	14	29.7	16.3	28.0	37.9	23.3	34.2
4	Domain knowledge	15	35.1	21.0	34.3	31.9	20.3	31.0
5	Mathematical reasoning	4	24.3	15.7	22.8	26.2	17.1	24.9
6	Weak clue	47	30.4	15.8	27.7	36.4	22.5	33.9
Overall			34.3	22.6	32.1	36.8	25.0	34.7

3) The terms used in the title should be derived from the original weibo.

4) Abbreviations are allowed. For example, "中国银行(Bank of China Limited)" can be abbreviated as "中行(China Bank)".

5) Due to length restrictions, some weibos may not be complete. For such cases, annotators are requested to write the title only according to existing content.

We further analyze two neural headline generation systems based on the manual headlines. From Table 10 we found the following facts. First, for the weak clue cases, evaluated with more reasonable reference headlines, both systems obtain a higher ROUGE score. Moreover, MRT still performs better than MLE which indicates that by considering the sentence-level loss, MRT could consistently outperform a word-level training strategy. Second, for the domain knowledge cases, MRT performs better than MLE with regard to manual reference headlines. We will show an example in Table 11 to analyze the reason. From the example, we find that the MLE system tends to extract a small piece of the original article to take as the headline. The MRT system, on the other hand, is able to output more reasonable headlines. Although the output of the MRT system has similar semantic meaning with the original headline, it has a lower ROUGE score than MLE. However, the MRT system outperforms the MLE system with manual references, which further proves that the MRT system is consistently better than the MLE system.

5.3 Error Analysis

We now turn to analyze what types of errors the MRT system is making. After carefully analyzing the system output with regard to the 100 examples, we classify the errors into five major categories.

Missing Salient Information Errors. They are the most serious errors, contributing to 63% of the overall errors. Missing salient information errors occur when a system generated headline fails to include the salient content of the original article. This type of errors can be attributed to a system's failure to capture the salient information of the original article.

Evaluation Errors. They are a major type of errors that contribute to 26% of the overall errors. An evaluation error occurs when a system outputs a headline that is semantically equivalent to a gold reference, but is scored low by ROUGE because of its failure to recognize that the generated headline and the corresponding gold reference are semantically equivalent. In other words, an evaluation error is not an error made by a system, but an error due to the naivety of ROUGE.

Repeated Words Errors. They contribute to 8% of the overall errors. They occur when a system correctly identifies the key content of the original article, but outputs multiple times the exact same word. This type of errors can be attributed to a system's failure to remember past generated words.

Extra Words Errors. They occur when a system outputs a word that has no relation with the original article, and they contribute to 11% of the overall errors.

Word Order Errors. They indicate that the system generated headline basically can cover most important content, but expresses different meanings due to the wrong word order. This type of errors contributes to 4% of the overall errors.

6 Related Work

Headline generation is a well-defined task standardized in DUC2003 and DUC2004. Various approaches have been proposed for headline generation: rule-based, statistical-based, and neural-based.

The rule-based models create a headline for a news article using handcrafted and linguistically motivated rules to guide the choice of a potential headline. Hedge trimmer[1] is a representative example of this approach which creates a headline by removing constituents from the parse tree of the first sentence until it reaches a specific length limit. Statistical-based methods make use of large-scale training data to learn correlations between words in headlines and articles[28]. The best system on DUC2004, TOPIARY[32] combines both linguistic and

Table 10. Performance of Domain Knowledge and Weak Clue Cases with Regard to Original References and Our Manual References

Category	Reference	MLE			MRT		
		$R1_{F1}$	$R2_{F1}$	RL_{F1}	$R1_{F1}$	$R2_{F1}$	RL_{F1}
Domain knowledge	Original	35.1	21.0	34.3	31.9	20.3	31.0
	Manual	36.5	23.8	35.7	43.1	29.2	41.7
Weak clue	Original	30.4	15.8	27.7	36.4	22.5	33.9
	Manual	41.8	30.2	40.3	44.2	32.1	42.6

statistical information to generate headlines. There are also studies that make use of knowledge bases to generate better headlines.

Table 11. Example from Domain Knowledge Cases

Article	胡茂华是腾讯第116号员工，历任腾讯总监、盛大CTO (旅游)、1号店技术副总裁。现担任云服务提供商多备份联合创始人&CEO。以下是他分享的创业经验：创业的第一道坎：做一个决定！创业的第二道坎：找搭档！创业的第三道坎：找人才.. Maohua Hu is the 116th staff of Tencent. He has been the director of Tencent, CTO of Shanda and VP of Technology of Yihaodian. Currently, he is the co-founder and CEO of Duobeifen, a cloud service provider company. The following is the start-up experience he shared: The first step of start-up is to make a decision. The second step of start-up is to find partners. The third step of start-up is to find talents ...
Original	多备份CEO胡茂华：创业路上的五道坎“游戏大观CEO of Duobeifen Maohua Hu: five steps on the way of start-up GameLook
Manual	胡茂华分享创业经验 Maohua Hu shared his start-up experience
MLE	做一个创业的第二道坎 The second step to do a start-up
MRT	胡茂华的创业经验 The start-up experience of Maohua Hu

Note: MLE and MRT represent the MLE system and the MRT system outputs respectively. Manual means one of the four manual reference headlines. Original means the original reference headlines.

With the advances of deep neural network, there are growing studies that design neural network for headline generation. [7] is the first to utilize attention-based encoder-decoder architecture in headline generation task. [2-4] explore the effectiveness of additional input features in neural headline generation systems. [5] provides LCSTS, a dataset for Chinese headline generation task. [27] proposes an online headline generation system that can read input and generate output at the same time. [3, 6, 22] try to incorporate the pointer network[23] into headline generation systems. [11] is targeted at the length controlling problem of headline generation. [12] attempts to accomplish the headline generation task in a semi-supervised fashion.

Headline generation is a type of summarization task in essence, and there are also many researches focusing on the neural network based summarization task. [33] proposes a hierarchical neural architecture to accomplish the task of extractive single document summarization. [34] jointly handles saliency ranking and relevance ranking in query-focused summarization with the strength of neural network. [35] explores a quite interesting task in which the system is required to generate a function name-like summary given a source code snippet.

7 Conclusions

In this paper, our contributions are as follows. 1) We gave a broad overview of existing approaches based on neural headline generation, with particular focus on how encoders, decoders or training strategies influence the overall performance of the neural headline generation systems. 2) We presented a quantitative analysis of recent neural headline generation systems and explored which factors benefit this task indeed. 3) We performed a detailed error analysis of typical models and datasets to explore the capability of the neural headline generation system.

To summarize, we observed several key factors that affect the performance of headline generation systems. 1) Adding more linguistic features would help to capture more complicated information of input articles. 2) Bi-directional recurrent neural networks perform better on modeling input articles. 3) Attention mechanism consistently benefit neural headline generation systems. 4) Copy mechanism is a promising method to expand the limited target vocabulary with regard to source input. 5) As a sentence-level training strategy, MRT could significantly outperform a word-level training strategy.

While the neural headline generation system is proved to be superior to all the other systems, our analysis also pointed out some aspects of neural headline generation systems that deserve further work, such as the handling of missing salient information, repeating words and extra words.

References

[1] Dorr B, Zajic D, Schwartz R. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proc. the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics on Text summarization workshop*, Volume 5, May 2003, pp.1-8.

[2] Chopra S, Auli M, Rush A M. Abstractive sentence summarization with attentive recurrent neural networks. In *Proc. the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2016, pp.93-98.

[3] Nallapati R, Zhou B, Santos C. Abstractive text summarization using sequence-to-sequence RNNS and beyond. http://aclweb.org/anthology/K/K16/K16-1028.pdf, May 2017.

[4] Takase S, Suzuki J, Okazaki N, Hirao T, Nagata M. Neural headline generation on abstract meaning representation. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.1054-1059.

[5] Hu B, Chen Q, Zhu F. LCSTS: A large scale Chinese short text summarization dataset. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, September 2015, pp.1967-1972.

[6] Gu J, Lu Z, Li H, Li V O. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.1631-1640.

[7] Rush A M, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, September 2015, pp.379-389.

[8] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is diffcult. *IEEE Transactions on Neural Networks*, 1994, 5(2): 157-166.

[9] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, October 2014, pp.1724-1734.

[10] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*, 1997, 9(8): 1735-1780.

[11] Kikuchi Y, Neubig G, Sasano R, Takamura H, Okumura M. Controlling output length in neural encoder-decoders. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.1328-1338.

[12] Miao Y, Blunsom P. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.319-328.

[13] Schuster M, Paliwal K K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997, 45(11): 2673-2681.

[14] Bengio Y, Ducharme R, Vincent P, Jauvin C. A neural probabilistic language model. *The Journal of Machine Learning Research*, 2003, 3: 1137-1155.

[15] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, May 2015.

[16] Shen S, Cheng Y, He Z, He W, Wu H, Sun M, Liu Y. Minimum risk training for neural machine translation. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.1683-1692.

[17] Ranzato M, Chopra S, Auli M, Zaremba W. Sequence level training with recurrent neural networks. In *Proc. ICLR*, May 2016.

[18] Och F J. Minimum error rate training in statistical machine translation. In *Proc. the 41st Annual Meeting on Association for Computational Linguistics*, July 2003, pp.160-167.

[19] Smith D A, Eisner J. Minimum risk annealing for training log-linear models. In *Proc. the COLING/ACL Main Conference Poster Sessions*, July 2006, pp.787-794.

[20] Gao J, He X, Yih W, Deng L. Learning continuous phrase representations for translation modeling. In *Proc. the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

[21] Lin C Y. ROUGE: A package for automatic evaluation of summaries. In *Proc. the Workshop on Text Summarization Branches Out*, July 2004.

[22] Gulcehre C, Ahn S, Nallapati R, Zhou B, Bengio Y. Pointing the unknown words. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.140-149.

[23] Vinyals O, Fortunato M, Jaitly N. Pointer networks. In *Proc. Advances in Neural Information Processing Systems*, Dec. 2015, pp.2692-2700.

[24] Jean S, Cho K, Memisevic R, Bengio Y. On using very large target vocabulary for neural machine translation. In *Proc. the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, July 2015, pp.1-10.

[25] Vogel S, Ney H, Tillmann C. HMM-based word alignment in statistical translation. In *Proc. the 16th Conference on Computational Linguistics*, Aug. 1996, pp.836-841.

[26] Tillmann C, Vogel S, Ney H, Zubiaga A. A DP-based search using monotone alignments in statistical translation. In *Proc. the 35th Annual Meeting of the Association for Computational Linguistics*, July 1997, pp.289-296.

[27] Yu L, Buys J, Blunsom P. Online segment to segment neural transduction. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.1307-1316.

[28] Banko M, Mittal V O, Witbrock M J. Headline generation based on statistical translation. In *Proc. the 38th Annual Meeting of the Association for Computational Linguistics*, Oct. 2000, pp.318-325.

[29] Napoles C, Gormley M, van Durme B. Annotated Gigaword. In *Proc. the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*, June 2012, pp.95-100.

[30] Zeiler M D. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*, 2012. https://arxiv.org/abs/1212.5701, May 2017.

[31] Luong T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, September 2015, pp.1412-1421.

[32] Zajic D, Dorr B, Schwartz R. BBN/UMD at DUC-2004: Topiary. In *Proc. the HLT-NAACL Document Understanding Workshop*, Jan. 2004, pp.112-119.

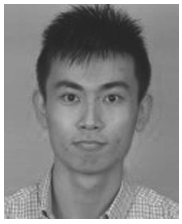
[33] Cheng J, Lapata M. Neural summarization by extracting sentences and words. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, Aug. 2016.

[34] Cao Z, Li W, Li S, Wei F, Li Y. AttSum: Joint learning of focusing and summarization with neural attention. In *Proc. the 26th International Conference on Computational Linguistics*, December 2016, pp.547-556.

[35] Allamanis M, Peng H, Sutton C. A convolutional attention network for extreme summarization of source code. In *Proc. the 33rd International Conference on Machine Learning*, June 2016, pp.2091-2100.

Ayana is a Ph.D. student of the Department of Computer Science and Technology, Tsinghua University, Beijing. She got her B.E. degree in computer science from the College of Computer Science and Technology, Inner Mongolia University, Hohhot, in 2006, and got her M.E. degree in 2009 from the College of Computer Science and Technology, Inner Mongolia University, Hohhot. Her research interest is document summarization.

Shi-Qi Shen is a Ph.D. student of the Department of Computer Science and Technology, Tsinghua University, Beijing. He got his B.E. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2012. His research interests are in the area of machine translation and deep learning for natural language processing.



Yan-Kai Lin is a Ph.D. student of the Department of Computer Science and Technology, Tsinghua University, Beijing. He got his B.E. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2014. His research interest is knowledge graph.

Cun-Chao Tu is a Ph.D. student of the Department of Computer Science and Technology, Tsinghua University, Beijing. He got his B.E. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2013. His research interests are user representation and social computation.

Yu Zhao is now working at IBM China Research Lab. He got his B.E. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2010, and he got his Ph.D. degree in computer science from Department of Computer Science and Technology, Tsinghua University, Beijing, in 2016. His main research interests include semantic analysis and representation learning. He has experiences in the fields of text classification, compositional semantics, and entity linking.

Zhi-Yuan Liu is an assistant researcher of the Department of Computer Science and Technology, Tsinghua University, Beijing. He got his B.E. degree and Ph.D. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2006 and 2011 respectively. His research interests are natural language processing and social computation. He has published over 40 papers in international journals and conferences including ACM Transactions, IJCAI, AAAI, ACL and EMNLP. He was awarded Tsinghua Excellent Doctoral Dissertation in 2011, and obtained Excellent Doctoral Dissertation awarded by Chinese Association for Artificial Intelligence in 2012, and Excellent Post-doctoral Fellow Award at Tsinghua University in 2013.

Mao-Song Sun is a professor at the Department of Computer Science and Technology in Tsinghua University, Beijing. He received his Ph.D. degree in computational linguistics from City University of Hong Kong, Hong Kong, in 2004. His research interests include natural language processing, Web intelligence, and machine learning.