Searching for Effective Neural Extractive Summarization: What Works and What's Next

Ming Zhong[*], **Pengfei Liu**[*], **Danqing Wang**, **Xipeng Qiu**[†], **Xuanjing Huang**
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{mzhong18,pfliu14,dqwang18,xpqiu,xjhuang}@fudan.edu.cn

Abstract

The recent years have seen remarkable success in the use of deep neural networks on text summarization. However, there is no clear understanding of *why* they perform so well, or *how* they might be improved. In this paper, we seek to better understand how neural extractive summarization systems could benefit from different types of model architectures, transferable knowledge and learning schemas. Additionally, we find an effective way to improve current frameworks and achieve the state-of-the-art result on CNN/DailyMail by a large margin based on our observations and analyses. Hopefully, our work could provide more clues for future research on extractive summarization. Source code will be available on Github[1].

1 Introduction

Recent years has seen remarkable success in the use of deep neural networks for text summarization (See et al., 2017; Celikyilmaz et al., 2018; Jadhav and Rajan, 2018). So far, most research utilizing the neural network for text summarization has revolved around architecture engineering (Zhou et al., 2018; Chen and Bansal, 2018; Gehrmann et al., 2018).

Despite their success, it remains poorly understood why they perform well and what their shortcomings are, which limits our ability to design better architectures. The rapid development of neural architectures calls for a detailed empirical study of analyzing and understanding existing models. In this paper, we primarily focus on extractive summarization since they are computationally efficient, and can generate grammatically and coherent summaries (Nallapati et al., 2017). and seek to better understand how neural network-based approaches to this task could benefit from different types of model architectures, transferable knowledge, and learning schemas, and how they might be improved.

Architectures Architecturally, the better performance usually comes at the cost of our understanding of the system. To date, we know little about the functionality of each neural component and the differences between them (Peters et al., 2018b), which raises the following typical questions: 1) How does the choice of different neural architectures (CNN, RNN, Transformer) influence the performance of the summarization system? 2) Which part of components matters for specific dataset? 3) Do current models suffer from the over-engineering problem?

Understanding the above questions can not only help us to choose suitable architectures in different application scenarios, but motivate us to move forward to more powerful frameworks.

External Transferable Knowledge and Learning schemas Clearly, the improvement in accuracy and performance is not merely because of the shift from feature engineering to structure engineering, but the flexible ways to incorporate external knowledge (Mikolov et al., 2013; Peters et al., 2018a; Devlin et al., 2018) and learning schemas to introduce extra instructive constraints (Paulus et al., 2017; Arumae and Liu, 2018). For this part, we make some first steps toward answers to the following questions: 1) Which type of pre-trained models (supervised or unsupervised pre-training) is more friendly to the summarization task? 2) When architectures are explored exhaustively, can we push the state-of-the-art results to a new level by introducing external transferable knowledge or changing another learning schema?

To make a comprehensive study of above an-

[*] These two authors contributed equally.
[†] Corresponding author.
[1] https://github.com/fastnlp/fastNLP

Perspective		Content	Sec.ID
Learning Schemas		Sup. & Reinforce.	4.4
Structure	Dec.	Pointer & SeqLab.	4.3.1
	Enc.	LSTM & Transformer	4.3.2
Knowledge	Exter.	GloVe BERT NEWS.	4.3.3
	Inter.	Random	

Table 1: Outline of our experimental design. Dec. and Enc. represent decoder and encoder respectively. Sup. denotes supervised learning and NEWS. means supervised pre-training knowledge.

alytical perspectives, we first build a testbed for summarization system, in which training and testing environment will be constructed. In the training environment, we design different summarization models to analyze how they influence the performance. Specifically, these models differ in the types of **architectures** (Encoders: CNN, LSTM, Transformer (Vaswani et al., 2017); Decoders: auto-regressive[2], non auto-regressive), **external transferable knowledge** (GloVe (Pennington et al., 2014), BERT (Devlin et al., 2018), NEWSROOM (Grusky et al., 2018)) and different **learning schemas** (supervised learning and reinforcement learning).

To peer into the internal working mechanism of above testing cases, we provide sufficient evaluation scenarios in the testing environment. Concretely, we present a multi-domain test, sentence shuffling test, and analyze models by different metrics: repetition, sentence length, and position bias, which we additionally developed to provide a better understanding of the characteristics of different datasets.

Empirically, our main observations are summarized as:

1) Architecturally speaking, models with auto-regressive decoder are prone to achieving better performance against non auto-regressive decoder. Besides, LSTM is more likely to suffer from the architecture overfitting problem while Transformer is more robust.

2) The success of extractive summarization system on the CNN/DailyMail corpus heavily relies on the ability to learn positional information of the sentence.

3) Unsupervised transferable knowledge is more useful than supervised transferable knowledge since the latter one is easily influenced by the domain shift problem.

4) We find an effective way to improve the current system, and achieving the state-of-the-art result on CNN/DailyMail by a large margin with the help of unsupervised transferable knowledge (**42.39** R-1 score). And this result can be further enhanced by introducing reinforcement learning (**42.69** R-1 score).

Hopefully, this detailed empirical study can provide more hints for the follow-up researchers to design better architectures and explore new state-of-the-art results along a right direction.

2 Related Work

The work is connected to the following threads of work of NLP research.

Task-oriented Neural Networks Interpreting Without knowing the internal working mechanism of the neural network, it is easy for us to get into a hobble when the performance of a task has reached the bottleneck. More recently, Peters et al. (2018b) investigate how different learning frameworks influence the properties of learned contextualized representations. Different from this work, in this paper, we focus on dissecting the neural models for text summarization.

A similar work to us is Kedzie et al. (2018), which studies how deep learning models perform context selection in terms of several typical summarization architectures, and domains. Compared with this work, we make a more comprehensive study and give more different analytic aspects. For example, we additionally investigate how transferable knowledge influence extractive summarization and a more popular neural architecture, Transformer. Besides, we come to inconsistent conclusions when analyzing the auto-regressive decoder. More importantly, our paper also shows how existing systems can be improved, and we have achieved a state-of-the-art performance on CNN/DailyMail.

Extractive Summarization Most of recent work attempt to explore different neural components or their combinations to build an end-to-end learning model. Specifically, these work instantiate their encoder-decoder framework by choosing recurrent neural networks (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhou et al., 2018) as encoder, auto-regressive decoder (Chen and

[2]Auto-regressive indicates that the decoder can make current prediction with knowledge of previous predictions.

Bansal, 2018; Jadhav and Rajan, 2018; Zhou et al., 2018) or non auto-regressive decoder (Isonuma et al., 2017; Narayan et al., 2018; Arumae and Liu, 2018) as decoder, based on pre-trained word representations (Mikolov et al., 2013; Pennington et al., 2014). However, how to use Transformer in extractive summarization is still a missing issue. In addition, some work uses reinforcement learning technique (Narayan et al., 2018; Wu and Hu, 2018; Chen and Bansal, 2018), which can provide more direct optimization goals. Although above work improves the performance of summarization system from different perspectives, yet a comprehensive study remains missing.

3 A Testbed for Text Summarization

To analyze neural summarization system, we propose to build a *Training-Testing* environment, in which different text cases (models) are firstly generated under different training settings, and they are further evaluated under different testing settings. Before the introduction of our Train-Testing testbed, we first give a description of text summarization.

3.1 Task Description

Existing methods of extractive summarization directly choose and output the salient sentences (or phrases) in the original document. Formally, given a document $D = d_1, \cdots, d_n$ consisting of n sentences, the objective is to extract a subset of sentences $R = r_1, \cdots, r_m$ from D, m is deterministic during training while is a hyper-parameter in testing phase. Additionally, each sentence contains $|d_i|$ words $d_i = x_1, \cdots, x_{|d_i|}$.

Generally, most of existing extractive summarization systems can be abstracted into the following framework, consisting of three major modules: **sentence encoder**, **document encoder** and **decoder**. At first, a sentence encoder will be utilized to convert each sentence d_i into a sentential representation $\mathbf{d}_i$. Then these sentence representations will be contextualized by a document encoder to $\mathbf{s}_i$. Finally, a decoder will extract a subset of sentences based on these contextualized sentence representations.

3.2 Setup for Training Environment

The objective of this step is to provide typical and diverse testing cases (models) in terms of model architectures, transferable knowledge and learning schemas.

3.2.1 Sentence Encoder

We instantiate our sentence encoder with CNN layer (Kim, 2014). We don't explore other options as sentence encoder since strong evidence of previous work (Kedzie et al., 2018) shows that the differences of existing sentence encoder don't matter too much for final performance.

3.2.2 Document Encoder

Given a sequence of sentential representation $\mathbf{d}_1, \cdots, \mathbf{d}_n$, the duty of document encoder is to contextualize each sentence therefore obtaining the contextualized representations $\mathbf{s}_1, \cdots, \mathbf{s}_n$. To achieve this goal, we investigate the LSTM-based structure and the Transformer structure, both of which have proven to be effective and achieved the state-of-the-art results in many other NLP tasks. Notably, to let the model make the best of its structural bias, stacking deep layers is allowed.

LSTM Layer Long short-term memory network (LSTM) was proposed by (Hochreiter and Schmidhuber, 1997) to specifically address this issue of learning long-term dependencies, which has proven to be effective in a wide range of NLP tasks, such as text classification (Liu et al., 2017, 2016b), semantic matching (Rocktäschel et al., 2015; Liu et al., 2016a), text summarization (Rush et al., 2015) and machine translation (Sutskever et al., 2014).

Transformer Layer Transformer (Vaswani et al., 2017) is essentially a feed-forward self-attention architecture, which achieves pairwise interaction by attention mechanism. Recently, Transformer has achieved great success in many other NLP tasks (Vaswani et al., 2017; Dai et al., 2018), and it is appealing to know how this neural module performs on text summarization task.

3.2.3 Decoder

Decoder is used to extract a subset of sentences from the original document based on contextualized representations: $\mathbf{s}_1, \cdots, \mathbf{s}_n$. Most existing architecture of decoders can divide into auto-regressive and non auto-regressive versions, both of which are investigated in this paper.

Sequence Labeling (SeqLab) The models, which formulate extractive summarization task as a sequence labeling problem, are equipped with non auto-regressive decoder. Formally, given a document D consisting of n sentences $d_1, \cdots, d_n$, the summaries are extracted by predicting a sequence of label $y_1, \cdots, y_n$ ($y_i \in \{0, 1\}$) for the document, where $y_i = 1$ represents the i-th sentence in the document should be included in the summaries.

Pointer Network (Pointer) As a representative of auto-regressive decoder, pointer network-based decoder has shown superior performance for extractive summarization (Chen and Bansal, 2018; Jadhav and Rajan, 2018). Pointer network selects the sentence by attention mechanism using *glimpse* operation (Vinyals et al., 2015). When it extracts a sentence, pointer network is aware of previous predictions.

3.2.4 External transferable knowledge

The success of neural network-based models on NLP tasks cannot only be attributed to the shift from feature engineering to structural engineering, but the flexible ways to incorporate external knowledge (Mikolov et al., 2013; Peters et al., 2018a; Devlin et al., 2018). The most common form of external transferable knowledge is the parameters pre-trained on other corpora.

To investigate how different pre-trained models influence the summarization system, we take the following pre-trained knowledge into consideration.

Unsupervised transferable knowledge Two typical unsupervised transferable knowledge are explored in this paper: context independent word embeddings (Mikolov et al., 2013; Pennington et al., 2014) and contextualized word embeddings (Peters et al., 2018a; Devlin et al., 2018), have put the state-of-the-art results to new level on a large number of NLP taks recently.

Supervised pre-trained knowledge Besides unsupervised pre-trained knowledge, we also can utilize parameters of networks pre-trained on other summarization datasets. The value of this investigation is to know transferability between different dataset. To achieve this, we first pre-train our model on the NEWSROOM dataset (Grusky et al., 2018), which is one of the largest datasets and contains samples from different domains. Then, we fine-tune our model on target domains that we investigate.

3.2.5 Learning Schemas

Utilizing external knowledge provides a way to seek new state-of-the-art results from the perspective of introducing extra data. Additionally, an alternative way is resorting to change the learning schema of the model. In this paper, we also explore how different learning schemas influence extractive summarization system by comparing supervised learning and reinforcement learning.

3.3 Setup for Testing Environment

In the testing environment, we provide sufficient evaluation scenarios to get the internal working mechanism of testing models. Next, we will make a detailed deception.

ROUGE Following previous work in text summarization, we evaluate the performance of different architectures with the standard ROUGE-1, ROUGE-2 and ROUGE-L F_1 scores (Lin, 2004) by using pyrouge package[3].

Cross-domain Evaluation We present a multi-domain evaluation, in which each testing model will be evaluated on multi-domain datasets based on CNN/DailyMail and NEWSROOM. Detail of the multi-domain datasets is descried in Tab. 2.

Repetition We design repetition score to test how different architectures behave diversely on avoiding generating unnecessary lengthy and repeated information. We use the percentage of repeated n-grams in extracted summary to measure the word-level repetition, which can be calculated as:

$$\mathrm{REP_n} = \frac{\mathrm{CountUniq}(ngram)}{\mathrm{Count}(ngram)} \tag{1}$$

where Count is used to count the number of n-grams and Uniq is used to eliminate n-gram duplication. The closer the word-based repetition score is to 1, the lower the repeatability of the words in summary.

Positional Bias It is meaningful to study whether the ground truth distribution of the datasets is different and how it affects different architectures. To achieve this we design a positional bias to describe the uniformity of ground truth distribution in different datasets, which can be calcu-

[3]pypi.python.org/pypi/pyrouge/0.1.3

lated as:

$$\text{PosBias} = \sum_{i=1}^{k} -p(i)\log(p(i)) \qquad (2)$$

We divide each article into k parts (we choose $k = 30$ because articles from CNN/DailyMail and NEWSROOM have 30 sentences by average) and $p(i)$ denotes the probability that the first golden label is in part i of the articles.

Sentence Length Sentence length will affect different metrics to some extent. We count the average length of the k-th sentence extracted from different decoders to explore whether the decoder could perceive the length information of sentences.

Sentence Shuffling We attempt to explore the impact of sentence position information on different structures. Therefore, we shuffle the orders of sentences and observe the robustness of different architectures to out-of-order sentences.

4 Experiment

4.1 Datasets

Instead of evaluating model solely on a single dataset, we care more about how our testing models perform on different types of data, which allows us to know if current models suffer from the over-engineering problem.

Domains	Train	Valid	Test
CNN/DailyMail	287,227	13,368	11,490
NYTimes	152,981	16,490	16,624
WashingtonPost	96,775	10,103	10,196
FoxNews	78,795	8,428	8,397
TheGuardian	58,057	6,376	6,273
NYDailyNews	55,653	6,057	5,904
WSJ	49,968	5,449	5,462
USAToday	44,921	4,628	4,781

Table 2: Statistics of multi-domain datasets based on CNN/DailyMail and NEWSROOM.

CNN/DailyMail The CNN/DailyMail question answering dataset (Hermann et al., 2015) modified by (Nallapati et al., 2016) is commonly used for summarization. The dataset consists of online news articles with paired human-generated summaries (3.75 sentences on average). For the data prepossessing, we use the data with non-anonymized version as (See et al., 2017), which doesn't replace named entities.

NEWSROOM Recently, NEWSROOM is constructed by (Grusky et al., 2018), which contains 1.3 million articles and summaries extracted from 38 major news publications across 20 years. We regard this diversity of sources as a diversity of summarization styles and select seven publications with the largest number of data as different domains to do the cross-domain evaluation. Due to the large scale data in NEWSROOM, we also choose this dataset to do transfer experiment.

4.2 Training Settings

For different learning schemas, we utilize cross entropy loss function and reinforcement learning method close to Chen and Bansal (2018) with a small difference: we use the precision of ROUGE-1 as a reward for every extracted sentence instead of the F_1 value of ROUGE-L.

For context-independent word representations (GloVe, Word2vec), we directly utilize them to initialize our words of each sentence, which can be fine-tuned during the training phase.

For BERT, we truncate the article to 512 tokens and feed it to a feature-based BERT (without gradient), concatenate the last four layers and get a 128-dimensional token embedding after passing through a MLP.

4.3 Experimental Observations and Analysis

Next, we will show our findings and analyses in terms of architectures and external transferable knowledge.

4.3.1 Analysis of Decoders

We understand the differences between decoder *Pointer* and *SeqLab* by probing their behaviours in different testing environments.

Domains From Tab. 3, we can observe that models with pointer-based decoder are prone to achieving better performance against SeqLab-based decoder. Specifically, among these eight datasets, models with pointer-based decoder outperform SeqLab on six domains and achieves comparable results on the other two domains. For example, in "`NYTimes`", "`WashingtonPost`" and "`TheGuardian`" domains, Pointer surpasses SeqLab by at least 1.0 improvment (R-1).

Model		R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Dec.	**Enc.**	**CNN/DM (2/3)**			**NYTimes (2)**			**WashingtonPost (1)**			**Foxnews (1)**		
	Lead	40.11	17.64	36.32	28.75	16.10	25.16	22.21	11.40	19.41	54.20	46.60	51.89
	Oracle	55.24	31.14	50.96	52.17	36.10	47.68	42.91	27.11	39.42	73.54	65.50	71.46
SeqLab	LSTM	41.22	18.72	37.52	30.26	17.18	26.58	21.27	10.78	18.56	59.32	51.82	56.95
	Transformer	41.31	**18.85**	37.63	30.03	17.01	26.37	21.74	10.92	18.92	59.35	51.82	56.97
Pointer	LSTM	**41.56**	18.77	**37.83**	31.31	**17.28**	**27.23**	**24.16**	**11.84**	**20.67**	**59.53**	**51.89**	**57.08**
	Transformer	41.36	18.59	37.67	**31.34**	17.25	27.16	23.77	11.63	20.48	59.35	51.68	56.90
Dec.	**Enc.**	**TheGuardian (1)**			**NYDailyNews (1)**			**WSJ (1)**			**USAToday (1)**		
	Lead	22.51	7.69	17.78	45.26	35.53	42.70	39.63	27.72	36.10	29.44	18.92	26.65
	Oracle	41.08	21.49	35.80	73.99	64.80	72.09	57.15	43.06	53.27	47.17	33.40	44.02
SeqLab	LSTM	23.02	8.12	18.29	53.13	43.52	50.53	41.94	29.54	38.19	30.30	18.96	27.40
	Transformer	23.49	8.43	18.65	53.66	44.19	51.07	42.98	**30.22**	39.02	30.97	19.77	28.03
Pointer	LSTM	24.71	8.55	19.30	53.31	43.37	50.52	43.29	30.20	**39.12**	31.73	19.89	28.50
	Transformer	**24.86**	**8.66**	**19.45**	**54.30**	**44.70**	**51.67**	**43.30**	30.17	39.07	**31.95**	**20.11**	**28.78**

Table 3: Results of different architectures over different domains, where **Enc.** and **Dec.** represent document encoder and decoder respectively. Lead means to extract the first k sentences as the summary, usually as a competitive lower bound. Oracle represents the ground truth extracted by the greedy algorithm (Nallapati et al., 2017), usually as the upper bound. The number k in parentheses denotes k sentences are extracted during testing and choose lead-k as a lower bound for this domain. All the experiments use word2vec to obtain word representations.

We attempt to explain this difference from the following three perspectives.

Repetition For domains that need to extract multiple sentences as the summary (first two domains in Tab. 3), Pointer is aware of the previous prediction which makes it to reduce the duplication of n-grams compared to SeqLab. As shown in Fig. 1(a), models with Pointer always get higher repetition scores than models with SeqLab when extracting six sentences, which indicates that Pointer does capture word-level information from previous selected sentences and has positive effects on subsequent decisions.

Positional Bias For domains that only need to extract one sentence as the summary (last six domains in Tab. 3), Pointer still performs better than SeqLab. As shown in Fig. 1(b), *the performance gap between these two decoders grows as the positional bias of different datasets increases*. For example, from the Tab. 3, we can see in the domains with low-value positional bias, such as "`FoxNews(1.8)`", "`NYDailyNews(1.9)`", SeqLab achieves closed performance against Pointer. By contrast, the performance gap grows when processing these domains with high-value positional bias ("`TheGuardian(2.9)`", "`WashingtonPost(3.0)`"). Consequently, SeqLab is more sensitive to positional bias, which impairs its performance on some datasets.

Sentence length We find *Pointer shows the ability to capture sentence length information based on previous predictions*, while SeqLab doesn't. We can see from the Fig. 1(c) that models with Pointer tend to choose longer sentences as the first sentence and greatly reduce the length of the sentence in the subsequent extractions. In comparison, it seems that models with SeqLab tend to extract sentences with similar length. The ability allows Pointer to adaptively change the length of the extracted sentences, thereby achieving better performance regardless of whether one sentence or multiple sentences are required.

4.3.2 Analysis of Encoders

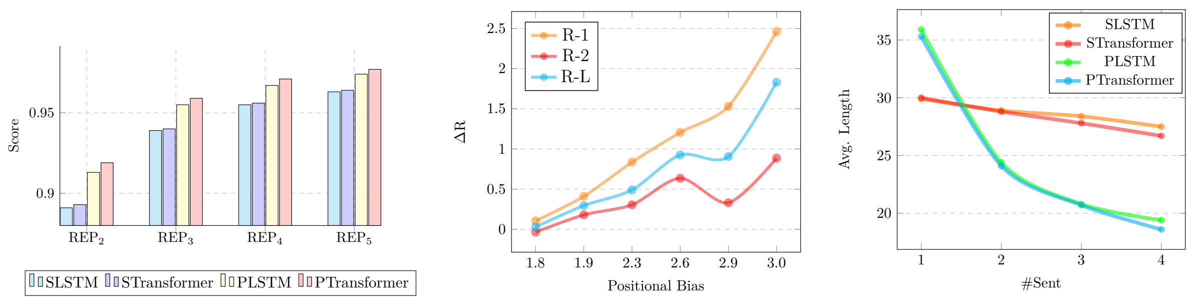
In this section, we make the analysis of two encoders LSTM and Transformer in different testing environments.

Domains From Tab. 3, we get the following observations:

1) Transformer can outperform LSTM on some datasets "`NYDailyNews`" by a relatively large margin while LSTM beats Transformer on some domains with closed improvements. Besides, during different training phases of these eight domains, the hyper-parameters of Transformer keep unchanged[4] while for LSTM, many sets of hyper-parameters are used[5].

[4] 4 layers 512 dimensions for Pointer and 12 layers 512 dimensions for SeqLab

[5] the number of layers searches in (2, 4, 6, 8) and dimen-



(a) Repetition score (b) Positional bias (c) Average length

Figure 1: Different behaviours of two decoders (SeqLab and Pointer) under different testing environment. (a) shows repetition scores of different architectures when extracting six sentences on CNN/DailyMail. (b) shows the relationship between ΔR and positional bias. The abscissa denotes the positional bias of six different datasets and ΔR denotes the average ROUGE difference between the two decoders under different encoders. (c) shows average length of k-th sentence extracted from different architectures.

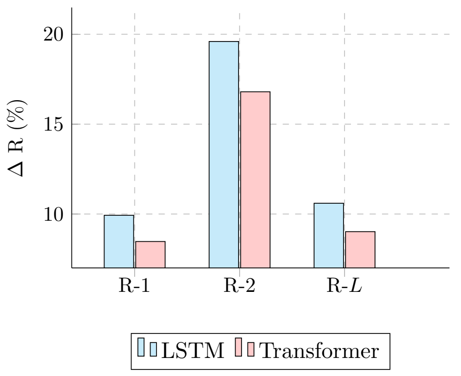


Figure 2: Results of different document encoders with Pointer on normal and shuffled CNN/DailyMail. ΔR denotes the decrease of performance when the sentences in document are shuffled.

α	β	R-1	R-2	R-L
1	0	37.90	15.69	34.31
$\sqrt{d}$	1	40.93	18.49	37.24
1	1	**41.31**	**18.85**	**37.63**
1	$\sqrt{d}$	40.88	18.42	37.19
0	1	40.39	17.67	36.54
Nallapati et al. (2017)		39.6	16.2	35.3
Narayan et al. (2018)		40.2	18.2	36.6

Table 4: Results of Transformer with SeqLab using different proportions of sentence embedding and positional embedding on CNN/DailyMail. The input of Transformer is $\alpha *$ sentence embedding plus $\beta *$ positional embedding[6]. The bottom half of the table contains models that have similar performance with Transformer that only know positional information.

Above phenomena suggest that LSTM easily suffers from the architecture overfitting problem compared with Transformer. Additionally, in our experimental setting, Transformer is more efficient to train since it is two or three times faster than LSTM.

2) When equipped with SeqLab decoder, Transformer always obtains a better performance compared with LSTM, the reason we think is due to the non-local bias (Wang et al., 2018) of Transformer.

Shuffled Testing In this settings, we shuffle the orders of sentences in training set while test set keeps unchanged. We compare two models with different encoders (LSTM, Transformer) and the results can be seen in Fig. 2. Generally, there is significant drop of performance about these two models. However, Transformer obtains lower decrease against LSTM, suggesting that Transformer are more robust.

Disentangling Testing Transformer provides us an effective way to disentangle position and content information, which enables us to design a specific experiment, investigating what role positional information plays.

As shown in Tab. 4, we dynamically regulate the ratio between sentence embedding and positional embedding by two coefficients α and β.

Surprisingly, we find even only utilizing positional embedding (the model is only told how many sentences the document contains), our model can achieve 40.08 on R-1, which is comparable to many existing models. By

sion searches in (512, 1024, 2048)

[6]In Vaswani et al. (2017), the input of Transformer is $\sqrt{d}$ $*$ word embedding plus positional embedding, so we design the above different proportions to carry out the disentangling test.

Model		R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Dec.	**Enc.**	**Baseline**			**+ GloVe**			**+ BERT**			**+ NEWSROOM**		
SeqLab	LSTM	41.22	18.72	37.52	**41.33**	**18.78**	**37.64**	42.18	19.64	38.53	41.48	**18.95**	37.78
	Transformer	41.31	**18.85**	37.63	40.19	18.67	37.51	42.28	**19.73**	38.59	41.32	18.83	37.63
Pointer	LSTM	**41.56**	18.77	**37.83**	41.15	18.38	37.43	**42.39**	19.51	**38.69**	41.35	18.59	37.61
	Transformer	41.36	18.59	37.67	41.10	18.38	37.41	42.09	19.31	38.41	**41.54**	18.73	**37.83**

Table 5: Results of different architectures with different pre-trained knowledge on CNN/DailyMail, where **Enc.** and **Dec.** represent document encoder and decoder respectively.

contrast, once the positional information is removed, the performance dropped by a large margin. This experiment shows that the success of such extractive summarization heavily relies on the ability of learning the positional information on CNN/DailyMail, which has been a benchmark dataset for most of current work.

4.3.3 Analysis of Transferable Knowledge

Next, we show how different types of transferable knowledge influences our summarization models.

Unsupervised Pre-training Here, as a baseline, *word2vec* is used to obtain word representations solely based on the training set of CNN/DailyMail.

As shown in Tab. 5, we can find that context-independent word representations can not contribute much to current models. However, when the models are equipped with BERT, we are excited to observe that the performances of all types of architectures are improved by a large margin. Specifically, the model `CNN-LSTM-Pointer` has achieved a new state-of-the-art with **42.11** on R-1, surpassing existing models dramatically.

Supervised Pre-training In most cases, our models can benefit from the pre-trained parameters learned from the NEWSROOM dataset. However, the model `CNN-LSTM-Pointer` fails and the performance are decreased. We understand this phenomenon by the following explanations: The transferring process from CNN/DailyMail to NEWSROOM suffers from the domain shift problem, in which the distribution of golden labels' positions are changed. And the observation from Fig. 2 shows that `CNN-LSTM-Pointer` is more sensitive to the ordering change, therefore obtaining a lower performance.

Why does BERT work? We investigate two different ways of using BERT to figure out from where BERT has brought improvement for extractive summarization system.

Models	R-1	R-2	R-L
Chen and Bansal (2018)	41.47	18.72	37.76
Dong et al. (2018)	41.50	18.70	37.60
Zhou et al. (2018)	41.59	19.01	37.98
Jadhav and Rajan (2018)[7]	41.60	18.30	37.70
LSTM + PN	41.56	18.77	37.83
LSTM + PN + RL	41.85	18.93	38.13
LSTM + PN + BERT	42.39	19.51	38.69
LSTM + PN + BERT + RL	**42.69**	**19.60**	**38.85**

Table 6: Evaluation on CNN/DailyMail. The top half of the table is currently state-of-the-art models, and the lower half is our models.

In the first usage, we feed each individual sentence to BERT to obtain sentence representation, which does not contain contextualized information, and the model gets a high R-1 score of 41.7. However, when we feed the entire article to BERT to obtain token representations and get the sentence representation through mean pooling, model performance soared to 42.3 R-1 score.

The experiment indicates that though BERT can provide a powerful sentence embedding, the key factor for extractive summarization is contextualized information and this type of information bears the positional relationship between sentences, which has been proven to be critical to extractive summarization task as above.

4.4 Learning Schema and Complementarity

Besides supervised learning, in text summarization, reinforcement learning has been recently used to introduce more constraints. In this paper, we also explore if several advanced techniques be complementary with each other.

We first choose the based model

[7]trained and evaluated on the anonymized version.

LSTM-Pointer and LSTM-Pointer + BERT, then the reinforcement learning are introduced aiming to further optimize our models. As shown in Tab. 6, we observe that even though the performance of LSTM+PN has been largely improved by BERT, when applying reinforcement learning, the performance can be improved further, which indicates that there is indeed a complementarity between architecture, transferable knowledge and reinforcement learning.

5 Conclusion

In this paper, we seek to better understand how neural extractive summarization systems could benefit from different types of model architectures, transferable knowledge, and learning schemas. Our detailed observations can provide more hints for the follow-up researchers to design more powerful learning frameworks.

Acknowledgment

We thank Jackie Chi Kit Cheung, Peng Qian for useful comments and discussions. We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by National Natural Science Foundation of China (No. 61751201 and 61672162), Shanghai Municipal Science and Technology Commission (16JC1420401 and 17JC1404100), Shanghai Municipal Science and Technology Major Project(No.2018SHZDZX01)and ZJLab.

References

Kristjan Arumae and Fei Liu. 2018. Reinforced extractive summarization with question-focused rewards. In *Proceedings of ACL 2018, Student Research Workshop*. pages 105–111.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. 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)*. volume 1, pages 1662–1675.

Yen-Chun Chen and Mohit Bansal. 2018. 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)*. volume 1, pages 675–686.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 484–494.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2018. Transformer-xl: Language modeling with longer-term dependency .

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 3739–3748.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 4098–4109.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. volume 1, pages 708–719.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2101–2110.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 142–151.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 1818–1828.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* .

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1034–1043.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*. pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1–10.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016* page 280.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. volume 1, pages 1747–1759.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. volume 1, pages 2227–2237.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 1499–1509.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. 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)*. volume 1, pages 1073–1083.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* .

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 7794–7803.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 654–663.