Extractive Summarization as Text Matching

Ming Zhong[*], Pengfei Liu[*], Yiran Chen, 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,yrchen19,dqwang18,xpqiu,xjhuang}@fudan.edu.cn

Abstract

This paper creates a *paradigm shift* with regard to the way we build neural extractive summarization systems. Instead of following the commonly used framework of extracting sentences individually and modeling the relationship between sentences, we formulate the extractive summarization task as a *semantic text matching* problem, in which a source document and candidate summaries will be (extracted from the original text) matched in a semantic space. Notably, this paradigm shift to semantic matching framework is *well-grounded* in our comprehensive analysis of the inherent gap between sentence-level and summary-level extractors based on the property of the dataset.

Besides, even instantiating the framework with a simple form of a matching model, we have driven the state-of-the-art extractive result on CNN/DailyMail to a new level (44.41 in ROUGE-1). Experiments on the other five datasets also show the effectiveness of the matching framework. We believe the power of this matching-based summarization framework has not been fully exploited. To encourage more instantiations in the future, we have released our codes, processed dataset, as well as generated summaries in https://github.com/maszhongming/MatchSum.

1 Introduction

The task of automatic text summarization aims to compress a textual document to a shorter highlight while keeping salient information on the original text. In this paper, we focus on extractive summarization since it usually generates semantically and grammatically correct sentences (Dong et al., 2018; Nallapati et al., 2017) and computes faster.

Currently, most of the neural extractive summarization systems score and extract sentences (or smaller semantic unit (Xu et al., 2019)) one by one from the original text, model the relationship between the sentences, and then select several sentences to form a summary. Cheng and Lapata (2016); Nallapati et al. (2017) formulate the extractive summarization task as a sequence labeling problem and solve it with an encoder-decoder framework. These models make independent binary decisions for each sentence, resulting in high redundancy. A natural way to address the above problem is to introduce an auto-regressive decoder (Chen and Bansal, 2018; Jadhav and Rajan, 2018; Zhou et al., 2018), allowing the scoring operations of different sentences to influence on each other. Trigram Blocking (Paulus et al., 2017; Liu and Lapata, 2019), as a more popular method recently, has the same motivation. At the stage of selecting sentences to form a summary, it will skip the sentence that has trigram overlapping with the previously selected sentences. Surprisingly, this simple method of removing duplication brings a remarkable performance improvement on CNN/DailyMail.

The above systems of modeling the relationship between sentences are essentially sentence-level extractors, rather than considering the semantics

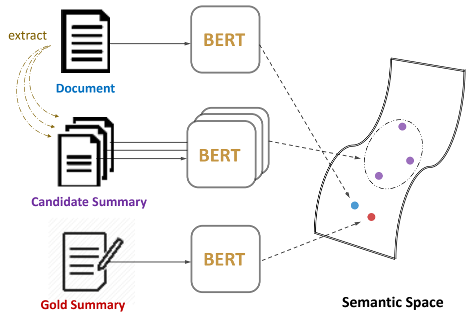


Figure 1: MATCHSUM framework. We match the contextual representations of the document with gold summary and candidate summaries (extracted from the document). Intuitively, better candidate summaries should be semantically closer to the document, while the gold summary should be the closest.

[*]These two authors contributed equally.
[†]Corresponding author.

of the entire summary. This makes them more inclined to select highly generalized sentences while ignoring the coupling of multiple sentences. Narayan et al. (2018b); Bae et al. (2019) utilize reinforcement learning (RL) to achieve summary-level scoring, but still limited to the architecture of sentence-level summarizers.

To better understand the advantages and limitations of sentence-level and summary-level approaches, we conduct an analysis on six benchmark datasets (in Section 3) to explore the characteristics of these two methods. We find that there is indeed an inherent gap between the two approaches across these datasets, which motivates us to propose the following summary-level method.

In this paper, we propose a novel summary-level framework (MATCHSUM, Figure 1) and conceptualize extractive summarization as a semantic text matching problem. The principle idea is that a good summary should be more semantically similar as a whole to the source document than the unqualified summaries. Semantic text matching is an important research problem to estimate semantic similarity between a source and a target text fragment, which has been applied in many fields, such as information retrieval (Mitra et al., 2017), question answering (Yih et al., 2013; Severyn and Moschitti, 2015), natural language inference (Wang and Jiang, 2016; Wang et al., 2017) and so on. One of the most conventional approaches to semantic text matching is to learn a vector representation for each text fragment, and then apply typical similarity metrics to compute the matching scores.

Specific to extractive summarization, we propose a Siamese-BERT architecture to compute the similarity between the source document and the candidate summary. Siamese BERT leverages the pre-trained BERT (Devlin et al., 2019) in a Siamese network structure (Bromley et al., 1994; Hoffer and Ailon, 2015; Reimers and Gurevych, 2019) to derive semantically meaningful text embeddings that can be compared using cosine-similarity. A good summary has the highest similarity among a set of candidate summaries.

We evaluate the proposed matching framework and perform significance testing on a range of benchmark datasets. Our model outperforms strong baselines significantly in all cases and improve the state-of-the-art extractive result on CNN/DailyMail. Besides, we design experiments to observe the gains brought by our framework.

We summarize our contributions as follows:

1) Instead of scoring and extracting sentences one by one to form a summary, we formulate extractive summarization as a semantic text matching problem and propose a novel summary-level framework. Our approach bypasses the difficulty of summary-level optimization by contrastive learning, that is, a good summary should be more semantically similar to the source document than the unqualified summaries.

2) We conduct an analysis to investigate whether extractive models must do summary-level extraction based on the property of dataset, and attempt to quantify the inherent gap between sentence-level and summary-level methods.

3) Our proposed framework has achieved superior performance compared with strong baselines on six benchmark datasets. Notably, we obtain *a state-of-the-art extractive result on CNN/DailyMail (44.41 in ROUGE-1) by only using the base version of BERT*. Moreover, we seek to observe where the performance gain of our model comes from.

2 Related Work

2.1 Extractive Summarization

Recent research work on extractive summarization spans a large range of approaches. These work usually instantiate their encoder-decoder framework by choosing RNN (Zhou et al., 2018), Transformer (Zhong et al., 2019b; Wang et al., 2019) or GNN (Wang et al., 2020) as encoder, non-auto-regressive (Narayan et al., 2018b; Arumae and Liu, 2018) or auto-regressive decoders (Jadhav and Rajan, 2018; Liu and Lapata, 2019). Despite the effectiveness, these models are essentially sentence-level extractors with individual scoring process favor the highest scoring sentence, which probably is not the optimal one to form summary[1].

The application of RL provides a means of summary-level scoring and brings improvement (Narayan et al., 2018b; Bae et al., 2019). However, these efforts are still limited to auto-regressive or non-auto-regressive architectures. Besides, in the non-neural approaches, the Integer Linear Programming (ILP) method can also be used for summary-level scoring (Wan et al., 2015).

In addition, there is some work to solve extractive summarization from a semantic perspective before this paper, such as concept coverage (Gillick

[1]We will quantify this phenomenon in Section 3.

and Favre, 2009), reconstruction (Miao and Blunsom, 2016) and maximize semantic volume (Yogatama et al., 2015).

2.2 Two-stage Summarization

Recent studies (Alyguliyev, 2009; Galanis and Androutsopoulos, 2010; Zhang et al., 2019a) have attempted to build two-stage document summarization systems. Specific to extractive summarization, the first stage is usually to extract some fragments of the original text, and the second stage is to select or modify on the basis of these fragments.

Chen and Bansal (2018) and Bae et al. (2019) follow a hybrid *extract-then-rewrite* architecture, with policy-based RL to bridge the two networks together. Lebanoff et al. (2019); Xu and Durrett (2019); Mendes et al. (2019) focus on the *extract-then-compress* learning paradigm, which will first train an extractor for content selection. Our model can be viewed as an *extract-then-match* framework, which also employs a sentence extractor to prune unnecessary information.

3 Sentence-Level or Summary-Level? A Dataset-dependent Analysis

Although previous work has pointed out the weakness of sentence-level extractors, there is no systematic analysis towards the following questions: 1) For extractive summarization, is the *summary-level extractor* better than the *sentence-level extractor*? 2) Given a dataset, which extractor should we choose based on the characteristics of the data, and what is the inherent gap between these two extractors?

In this section, we investigate the gap between sentence-level and summary-level methods on six benchmark datasets, which can instruct us to search for an effective learning framework. It is worth noting that the sentence-level extractor we use here doesn't include a redundancy removal process so that we can estimate the effect of the summary-level extractor on redundancy elimination. Notably, the analysis method to estimate the theoretical effectiveness presented in this section is generalized and can be applicable to any summary-level approach.

3.1 Definition

We refer to $D = \{s_1, \cdots, s_n\}$ as a single document consisting of n sentences, and $C = \{s_1, \cdots, s_k, |s_i \in D\}$ as a *candidate summary* including k ($k \leq n$) sentences extracted from a document. Given a document D with its *gold summary* C^*, we measure a *candidate summary* C by calculating the ROUGE (Lin and Hovy, 2003) value between C and C^* in two levels:

1) Sentence-Level Score:

$$\mathrm{g}^{sen}(C) = \frac{1}{|C|}\sum_{s \in C} \mathrm{R(s, C^*)}, \tag{1}$$

where s is the sentence in C and $|C|$ represents the number of sentences. $\mathrm{R}(\cdot)$ denotes the average ROUGE score[2]. Thus, $\mathrm{g}^{sen}(C)$ indicates the average overlaps between each sentence in C and the gold summary C^*.

2) Summary-Level Score:

$$\mathrm{g}^{sum}(C) = \mathrm{R}(C, C^*), \tag{2}$$

where $\mathrm{g}^{sum}(C)$ considers sentences in C as a whole and then calculates the ROUGE score with the gold summary C^*.

Pearl-Summary We define the *pearl-summary* to be the summary that has a lower sentence-level score but a higher summary-level score.

Definition 1 *A candidate summary C is defined as a* ***pearl-summary*** *if there exists another candidate summary C' that satisfies the inequality: $\mathrm{g}^{sen}(C') > \mathrm{g}^{sen}(C)$ while $\mathrm{g}^{sum}(C') < \mathrm{g}^{sum}(C)$.* Clearly, if a candidate summary is a pearl-summary, it is challenging for sentence-level summarizers to extract it.

Best-Summary The best-summary refers to a summary has highest summary-level score among all the candidate summaries.

Definition 2 *A summary $\hat{C}$ is defined as the* ***best-summary*** *when it satisfies: $\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, \mathrm{g}^{sum}(C)$, where $\mathcal{C}$ denotes all the candidate summaries of the document.*

3.2 Ranking of Best-Summary

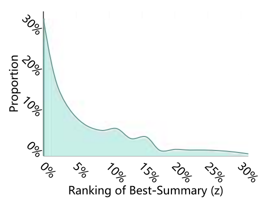
For each document, we sort all candidate summaries[3] in descending order based on the *sentence-level score*, and then define z as the rank index of the best-summary $\hat{C}$.

[2]Here we use mean F_1 of ROUGE-1, ROUGE-2 and ROUGE-L.

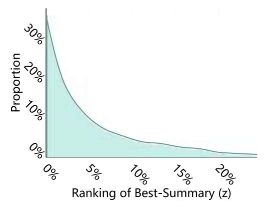
[3]We use an approximate method here: take #Ext (see Table 1) of ten highest-scoring sentences to form candidate summaries.

Datasets	**Source**	**Type**	**# Pairs** Train	Valid	Test	**# Tokens** Doc.	Sum.	**# Ext**
Reddit	Social Media	SDS	41,675	645	645	482.2	28.0	2
XSum	News	SDS	203,028	11,273	11,332	430.2	23.3	2
CNN/DM	News	SDS	287,084	13,367	11,489	766.1	58.2	3
WikiHow	Knowledge Base	SDS	168,126	6,000	6,000	580.8	62.6	4
PubMed	Scientific Paper	SDS	83,233	4,946	5,025	444.0	209.5	6
Multi-News	News	MDS	44,972	5,622	5,622	487.3	262.0	9

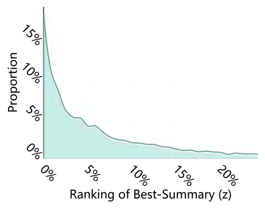
Table 1: Datasets overview. SDS represents single-document summarization and MDS represents multi-document summarization. The data in Doc. and Sum. indicates the average length of document and summary in the test set respectively. # Ext denotes the number of sentences should extract in different datasets.



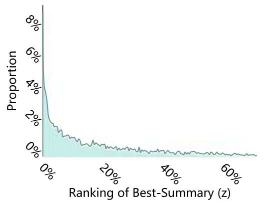
(a) Reddit



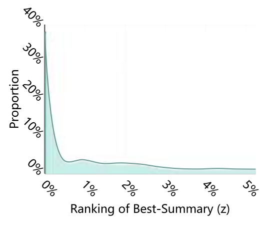
(b) XSum



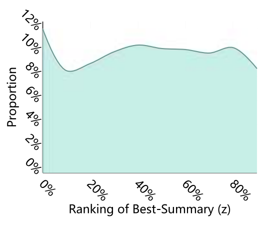
(c) CNN/DM



(d) WikiHow



(e) PubMed



(f) Multi-News

Figure 2: Distribution of $z(\%)$ on six datasets. Because the number of candidate summaries for each document is different (short text may have relatively few candidates), we use z / *number of candidate summaries* as the X-axis. The Y-axis represents the proportion of the best-summaries with this rank in the test set.

Intuitively, 1) if $z = 1$ ($\hat{C}$ comes first), it means that the best-summary is composed of sentences with the highest score; 2) If $z > 1$, then the best-summary is a pearl-summary. And as z increases ($\hat{C}$ gets lower rankings), we could find more candidate summaries whose sentence-level score is higher than best-summary, which leads to the learning difficulty for sentence-level extractors.

Since the appearance of the pearl-summary will bring challenges to sentence-level extractors, we attempt to investigate the proportion of pearl-summary in different datasets on six benchmark datasets. A detailed description of these datasets is displayed in Table 1.

As demonstrated in Figure 2, we can observe that for all datasets, most of the best-summaries are not made up of the highest-scoring sentences. Specifically, for `CNN/DM`, only 18.9% of best-summaries are not pearl-summary, indicating sentence-level extractors will easily fall into a local optimization, missing better candidate summaries.

Different from `CNN/DM`, `PubMed` is most suitable for sentence-level summarizers, because most of best-summary sets are not pearl-summary. Additionally, it is challenging to achieve good performance on `WikiHow` and `Multi-News` without a summary-level learning process, as these two datasets are most evenly distributed, that is, the appearance of pearl-summary makes the selection of the best-summary more complicated.

In conclusion, the proportion of the pearl-summaries in all the best-summaries is a property to characterize a dataset, which will affect our choices of summarization extractors.

3.3 Inherent Gap between Sentence-Level and Summary-Level Extractors

Above analysis has explicated that the summary-level method is better than the sentence-level method because it can pick out pearl-summaries, but how much improvement can it bring given a specific dataset?

Based on the definition of Eq. (1) and (2), we can characterize the upper bound of the sentence-level and summary-level summarization systems for a document D as:

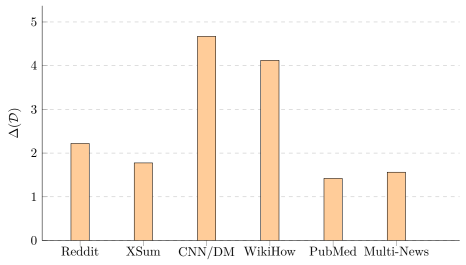


Figure 3: $\Delta(\mathcal{D})$ for different datasets.

$$\alpha^{sen}(D) = \max_{C \in \mathcal{C}_D} \mathrm{g}^{sen}(C), \qquad (3)$$

$$\alpha^{sum}(D) = \max_{C \in \mathcal{C}_D} \mathrm{g}^{sum}(C), \qquad (4)$$

where $\mathcal{C}_D$ is the set of candidate summaries extracted from D.

Then, we quantify the potential gain for a document D by calculating the difference between $\alpha^{sen}(D)$ and $\alpha^{sum}(D)$:

$$\Delta(D) = \alpha^{sum}(D) - \alpha^{sen}(D). \qquad (5)$$

Finally, a dataset-level potential gain can be obtained as:

$$\Delta(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \Delta(D), \qquad (6)$$

where $\mathcal{D}$ represents a specific dataset and $|\mathcal{D}|$ is the number of documents in this dataset.

We can see from Figure 3, the performance gain of the summary-level method varies with the dataset and has an improvement at a maximum 4.7 on CNN/DM. From Figure 3 and Table 1, we can find the performance gain is related to the length of reference summary for different datasets. In the case of short summaries (Reddit and XSum), the perfect identification of pearl-summaries does not lead to much improvement. Similarly, multiple sentences in a long summary (PubMed and Multi-News) already have a large degree of semantic overlap, making the improvement of the summary-level method relatively small. But for a medium-length summary (CNN/DM and WikiHow, about 60 words), the summary-level learning process is rewarding. We will discuss this performance gain with specific models in Section 5.4.

4 Summarization as Matching

The above quantitative analysis suggests that for most of the datasets, sentence-level extractors are inherently unaware of pearl-summary, so obtaining the best-summary is difficult. To better utilize the above characteristics of the data, we propose a summary-level framework which could score and extract a summary directly.

Specifically, we formulate the extractive summarization task as a semantic text matching problem, in which a source document and candidate summaries will be (extracted from the original text) matched in a semantic space. The following section will detail how we instantiate our proposed matching summarization framework by using a simple siamese-based architecture.

4.1 Siamese-BERT

Inspired by siamese network structure (Bromley et al., 1994), we construct a Siamese-BERT architecture to match the document D and the candidate summary C. Our Siamese-BERT consists of two BERTs with tied-weights and a cosine-similarity layer during the inference phase.

Unlike the modified BERT used in (Liu, 2019; Bae et al., 2019), we directly use the original BERT to derive the semantically meaningful embeddings from document D and candidate summary C since we need not obtain the sentence-level representation. Thus, we use the vector of the '`[CLS]`' token from the top BERT layer as the representation of a document or summary. Let $\mathbf{r}_D$ and $\mathbf{r}_C$ denote the embeddings of the document D and candidate summary C. Their similarity score is measured by $f(D, C) = \mathrm{cosine}(\mathbf{r}_D, \mathbf{r}_C)$.

In order to fine-tune Siamese-BERT, we use a margin-based triplet loss to update the weights. Intuitively, the gold summary C^* should be semantically closest to the source document, which is the first principle our loss should follow:

$$\mathcal{L}_1 = \max(0, f(D, C) - f(D, C^*) + \gamma_1), \qquad (7)$$

where C is the candidate summary in D and γ_1 is a margin value. Besides, we also design a pairwise margin loss for all the candidate summaries. We sort all candidate summaries in descending order of ROUGE scores with the gold summary. Naturally, the candidate pair with a larger ranking gap should have a larger margin, which is the second principle to design our loss function:

$$\begin{aligned}\mathcal{L}_2 = \max(0, f(D, C_j) - f(D, C_i) \\ + (j - i) * \gamma_2) \quad (i < j),\end{aligned} \qquad (8)$$

where C_i represents the candidate summary ranked i and γ_2 is a hyperparameter used to distinguish between good and bad candidate summaries. Finally, our margin-based triplet loss can be written as:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2. \tag{9}$$

The basic idea is to let the gold summary have the highest matching score, and at the same time, a better candidate summary should obtain a higher score compared with the unqualified candidate summary. Figure 1 illustrate this idea.

In the inference phase, we formulate extractive summarization as a task to search for the best summary among all the candidates $\mathcal{C}$ extracted from the document D.

$$\hat{C} = \arg\max_{C \in \mathcal{C}} f(D, C). \tag{10}$$

4.2 Candidates Pruning

Curse of Combination The matching idea is more intuitive while it suffers from combinatorial explosion problems. For example, how could we determine the size of the candidate summary set or should we score all possible candidates? To alleviate these difficulties, we propose a simple candidate pruning strategy.

Concretely, we introduce a *content selection module* to pre-select salient sentences. The module learns to assign each sentence a salience score and prunes sentences irrelevant with the current document, resulting in a pruned document $D^{'} = \{s^{'}_1, \cdots, s^{'}_{ext} | s^{'}_i \in D\}$.

Similar to much previous work on two-stage summarization, our content selection module is a parameterized neural network. In this paper, we use BERTSUM (Liu and Lapata, 2019) without trigram blocking (we call it BERTEXT) to score each sentence. Then, we use a simple rule to obtain the candidates: generating all combinations of *sel* sentences subject to the pruned document, and reorganize the order of sentences according to the original position in the document to form candidate summaries. Therefore, we have a total of $\binom{ext}{sel}$ candidate sets.

5 Experiment

5.1 Datasets

In order to verify the effectiveness of our framework and obtain more convicing explanations, we perform experiments on six divergent mainstream datasets as follows.

	Reddit	XSum	CNN/DM	Wiki	PubMed	M-News
Ext	5	5	5	5	7	10
Sel	1, 2	1, 2	2, 3	3, 4, 5	6	9
Size	15	15	20	16	7	9

Table 2: Details about the candidate summary for different datasets. *Ext* denotes the number of sentences after we prune the original document, *Sel* denotes the number of sentences to form a candidate summary and *Size* is the number of final candidate summaries.

`CNN/DailyMail` (Hermann et al., 2015) is a commonly used news summarization dataset modified by Nallapati et al. (2016). `PubMed` (Cohan et al., 2018) is collected from scientific papers. We modify this dataset by using the introduction section as the document and the abstract section as the corresponding summary. `WikiHow` (Koupaee and Wang, 2018) is a diverse dataset extracted from an online knowledge base. `XSum` (Narayan et al., 2018a) is a one-sentence summary dataset to answer the question "What is the article about?". `Multi-News` (Fabbri et al., 2019) is a multi-document news summarization dataset, we concatenate the source documents as a single input. `Reddit` (Kim et al., 2019) is a highly abstractive dataset collected from social media platform. We use the TIFU-long version of Reddit.

5.2 Implementation Details

We use the base version of BERT to implement our models in all experiments. Adam optimizer (Kingma and Ba, 2014) with warming-up is used and our learning rate schedule follows Vaswani et al. (2017) as:

$$\mathrm{lr} = 2\mathrm{e}^{-3} \cdot \min(\mathrm{step}^{-0.5}, \mathrm{step} \cdot \mathrm{wm}^{-1.5}), \tag{11}$$

where each step is a batch size of 32 and wm denotes warmup steps of 10,000. We choose $\gamma_1 = 0$ and $\gamma_2 = 0.01$. When $\gamma_1 {<} 0.05$ and $0.005 {<} \gamma_2 {<} 0.05$ they have little effect on performance, otherwise they will cause performance degradation. We use the validation set to save three best checkpoints during training, and record the performance of the best checkpoints on the test set. Importantly, all the experimental results listed in this paper are the average of three runs. To obtain a Siamese-BERT model on `CNN/DM`, we use 8 Tesla-V100-16G GPUs for about 30 hours of training.

For datasets, we remove samples with empty document or summary and truncate the document

Model	R-1	R-2	R-L
LEAD	40.43	17.62	36.67
ORACLE	52.59	31.23	48.87
MATCH-ORACLE	51.08	26.94	47.22
BANDITSUM (Dong et al., 2018)	41.50	18.70	37.60
NEUSUM (Zhou et al., 2018)	41.59	19.01	37.98
JECS (Xu and Durrett, 2019)	41.70	18.50	37.90
HIBERT (Zhang et al., 2019b)	42.37	19.95	38.83
PNBERT (Zhong et al., 2019a)	42.39	19.51	38.69
PNBERT + RL	42.69	19.60	38.85
BERTEXT† (Bae et al., 2019)	42.29	19.38	38.63
BERTEXT† + RL	42.76	19.87	39.11
BERTEXT (Liu, 2019)	42.57	19.96	39.04
BERTEXT + Tri-Blocking	43.23	20.22	39.60
BERTSUM* (Liu and Lapata, 2019)	43.85	20.34	39.90
BERTEXT (Ours)	42.73	20.13	39.20
BERTEXT + Tri-Blocking (Ours)	43.18	20.16	39.56
MATCHSUM (BERT-base)	44.22	20.62	40.38
MATCHSUM (RoBERTa-base)	**44.41**	**20.86**	**40.55**

Table 3: Results on CNN/DM test set. The model with * indicates that the large version of BERT is used. BERTEXT† add an additional Pointer Network compared to other BERTEXT in this table.

Model	R-1	R-2	R-L
Reddit			
BERTEXT (Num = 1)	21.99	5.21	16.99
BERTEXT (Num = 2)	23.86	5.85	19.11
MATCHSUM (Sel = 1)	22.87	5.15	17.40
MATCHSUM (Sel = 2)	24.90	5.91	20.03
MATCHSUM (Sel = 1, 2)	**25.09**	**6.17**	**20.13**
XSum			
BERTEXT (Num = 1)	22.53	4.36	16.23
BERTEXT (Num = 2)	22.86	4.48	17.16
MATCHSUM (Sel = 1)	23.35	4.46	16.71
MATCHSUM (Sel = 2)	24.48	4.58	18.31
MATCHSUM (Sel = 1, 2)	**24.86**	**4.66**	**18.41**

Table 4: Results on test sets of Reddit and XSum. Num indicates how many sentences BERTEXT extracts as a summary and Sel indicates the number of sentences we choose to form a candidate summary.

to 512 tokens, therefore ORACLE in this paper is calculated on the truncated datasets. Details of candidate summary for the different datasets can be found in Table 2.

5.3 Experimental Results

Results on CNN/DM As shown in Table 3, we list strong baselines with different learning approaches. The first section contains *LEAD*, *ORACLE* and *MATCH-ORACLE*[4]. Because we prune documents before matching, *MATCH-ORACLE* is relatively low.

We can see from the second section, although RL can score the entire summary, it does not lead to much performance improvement. This is probably because it still relies on the sentence-level summarizers such as Pointer network or sequence labeling models, which select sentences one by one, rather than distinguishing the semantics of different summaries as a whole. Trigram Blocking is a simple yet effective heuristic on CNN/DM, even better than all redundancy removal methods based on neural models.

[4]*LEAD* and *ORACLE* are common baselines in the summarization task. The former means extracting the first several sentences of a document as a summary, the latter is the groundtruth used in extractive models training. *MATCH-ORACLE* is the groundtruth used to train MATCHSUM.

Compared with these models, our proposed MATCHSUM has outperformed all competitors by a large margin. For example, it beats BERTEXT by 1.51 ROUGE-1 score when using BERT-base as the encoder. Additionally, even compared with the baseline with BERT-large pre-trained encoder, our model MATCHSUM (BERT-base) still perform better. Furthermore, when we change the encoder to RoBERTa-base (Liu et al., 2019), the performance can be further improved. We think the improvement here is because RoBERTa introduced 63 million English news articles during pretraining. The superior performance on this dataset demonstrates the effectiveness of our proposed matching framework.

Results on Datasets with Short Summaries `Reddit` and `XSum` have been heavily evaluated by abstractive summarizer due to their short summaries. Here, we evaluate our model on these two datasets to investigate whether MATCHSUM could achieve improvement when dealing with summaries containing fewer sentences compared with other typical extractive models.

When taking just one sentence to match the original document, MATCHSUM degenerates into a re-ranking of sentences. Table 4 illustrates that this degradation can still bring a small improvement (compared to BERTEXT (Num = 1), 0.88 ΔR-1 on `Reddit`, 0.82 ΔR-1 on `XSum`). However, when the number of sentences increases to two and summary-level semantics need to be taken into account, MATCHSUM can obtain a more re-

Model	WikiHow			PubMed			Multi-News		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
LEAD	24.97	5.83	23.24	37.58	12.22	33.44	43.08	14.27	38.97
ORACLE	35.59	12.98	32.68	45.12	20.33	40.19	49.06	21.54	44.27
MATCH-ORACLE	35.22	10.55	32.87	42.21	15.42	37.67	47.45	17.41	43.14
BERTEXT	30.31	8.71	28.24	41.05	14.88	36.57	45.80	16.42	41.53
+ 3gram-Blocking	30.37	8.45	28.28	38.81	13.62	34.52	44.94	15.47	40.63
+ 4gram-Blocking	30.40	8.67	28.32	40.29	14.37	35.88	45.86	16.23	41.57
MATCHSUM (BERT-base)	**31.85**	**8.98**	**29.58**	**41.21**	**14.91**	**36.75**	**46.20**	**16.51**	**41.89**

Table 5: Results on test sets of WikiHow, PubMed and Multi-News. MATCHSUM beats the state-of-the-art BERT model with Ngram Blocking on all different domain datasets.

markable improvement (compared to BERTEXT (Num = 2), 1.04 ΔR-1 on Reddit, 1.62 ΔR-1 on XSum).

In addition, our model maps candidate summary as a whole into semantic space, so it can flexibly choose any number of sentences, while most other methods can only extract a fixed number of sentences. From Table 4, we can see this advantage leads to further performance improvement.

Results on Datasets with Long Summaries When the summary is relatively long, summary-level matching becomes more complicated and is harder to learn. We aim to compare the difference between Trigram Blocking and our model when dealing with long summaries.

Table 5 presents that although Trigram Blocking works well on CNN/DM, it does not always maintain a stable improvement. Ngram Blocking has little effect on WikiHow and Multi-News, and it causes a large performance drop on PubMed. We think the reason is that Ngram Blocking cannot really understand the semantics of sentences or summaries, just restricts the presence of entities with many words to only once, which is obviously not suitable for the scientific domain where entities may often appear multiple times.

On the contrary, our proposed method does not have strong constraints but aligns the document with the summary from semantic space. Experiment results display that our model is robust on all domains, especially on WikiHow, MATCHSUM beats the state-of-the-art model by 1.54 R-1 score.

5.4 Analysis

Our analysis here is driven by two questions:

1) Whether the benefits of MATCHSUM are consistent with the property of the dataset analyzed in Section 3?

2) Why have our model achieved different performance gains on diverse datasets?

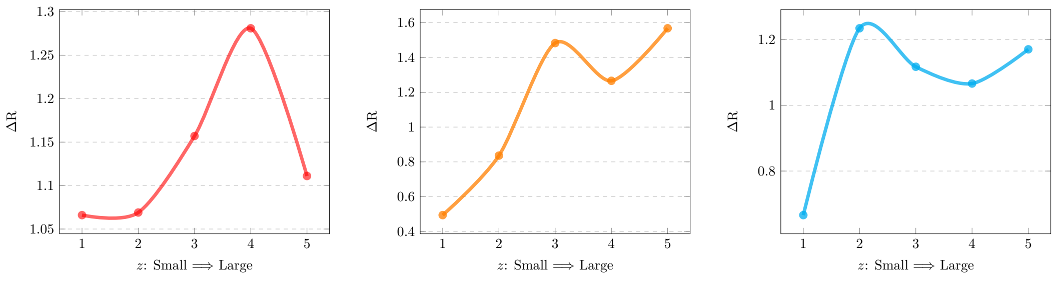
Dataset Splitting Testing Typically, we choose three datasets (XSum, CNN/DM and WikiHow) with the largest performance gain for this experiment. We split each test set into roughly equal numbers of five parts according to z described in Section 3.2, and then experiment with each subset.

Figure 4 shows that the performance gap between MATCHSUM and BERTEXT is always the smallest when the best-summary is not a pearl-summary ($z = 1$). The phenomenon is in line with our understanding, in these samples, the ability of the summary-level extractor to discover pearl-summaries does not bring advantages.

As z increases, the performance gap generally tends to increase. Specifically, the benefit of MATCHSUM on CNN/DM is highly consistent with the appearance of pearl-summary. It can only bring an improvement of 0.49 in the subset with the smallest z, but it rises sharply to 1.57 when z reaches its maximum value. WikiHow is similar to CNN/DM, when best-summary consists entirely of highest-scoring sentences, the performance gap is obviously smaller than in other samples. XSum is slightly different, although the trend remains the same, our model does not perform well in the samples with the largest z, which needs further improvement and exploration.

From the above comparison, we can see that the performance improvement of MATCHSUM is concentrated in the samples with more pearl-summaries, which illustrates our semantic-based summary-level model can capture sentences that are not particularly good when viewed individually, thereby forming a better summary.

Comparison Across Datasets Intuitively, improvements brought by MATCHSUM framework



(a) XSum (b) CNN/DM (c) WikiHow

Figure 4: Datasets splitting experiment. We split test sets into five parts according to z described in Section 3.2. The X-axis from left to right indicates the subsets of the test set with the value of z from small to large, and the Y-axis represents the ROUGE improvement of MATCHSUM over BERTEXT on this subset.

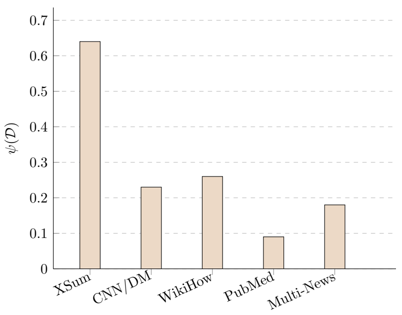


Figure 5: ψ of different datasets. Reddit is excluded because it has too few samples in the test set.

should be associated with inherent gaps presented in Section 3.3. To better understand their relation, we introduce $\Delta(\mathcal{D})^*$ as follows:

$$\Delta(D)^* = \mathrm{g}^{sum}(C_{MS}) - \mathrm{g}^{sum}(C_{BE}), \qquad (12)$$

$$\Delta(\mathcal{D})^* = \frac{1}{|\mathcal{D}|}\sum_{D\in\mathcal{D}} \Delta(D)^*, \qquad (13)$$

where C_{MS} and C_{BE} represent the candidate summary selected by MATCHSUM and BERTEXT in the document D, respectively. Therefore, $\Delta(\mathcal{D})^*$ can indicate the improvement by MATCHSUM over BERTEXT on dataset $\mathcal{D}$. Moreover, compared with the inherent gap between sentence-level and summary-level extractors, we define the ratio that MATCHSUM can learn on dataset $\mathcal{D}$ as:

$$\psi(\mathcal{D}) = \Delta(\mathcal{D})^*/\Delta(\mathcal{D}), \qquad (14)$$

where $\Delta(\mathcal{D})$ is the inherent gap between sentence-level and summary-level extractos.

It is clear from Figure 5, the value of $\psi(\mathcal{D})$ depends on z (see Figure 2) and the length of the gold summary (see Table 1). As the gold summaries get longer, the upper bound of summary-level approaches becomes more difficult for our model to reach. MATCHSUM can achieve 0.64 $\psi(\mathcal{D})$ on `XSum` (23.3 words summary), however, $\psi(\mathcal{D})$ is less than 0.2 in `PubMed` and `Multi-News` whose summary length exceeds 200. From another perspective, when the summary length are similar, our model performs better on datasets with more pearl-summaries. For instance, z is evenly distributed in `Multi-News` (see Figure 2), so higher $\psi(\mathcal{D})$ (0.18) can be obtained than `PubMed` (0.09), which has the least pearl-summaries.

A better understanding of the dataset allows us to get a clear awareness of the strengths and limitations of our framework, and we also hope that the above analysis could provide useful clues for future research on extractive summarization.

6 Conclusion

We formulate the extractive summarization task as a semantic text matching problem and propose a novel summary-level framework to match the source document and candidate summaries in the semantic space. We conduct an analysis to show how our model could better fit the characteristic of the data. Experimental results show MATCHSUM outperforms the current state-of-the-art extractive model on six benchmark datasets, which demonstrates the effectiveness of our method.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831103), National Natural Science Foundation of China (No. U1936214 and 61672162), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

References

RM Alyguliyev. 2009. The two-stage unsupervised approach to multidocument summarization. *Automatic Control and Computer Sciences*, 43(5):276.

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

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang-goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 10–20.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

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.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 615–621.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *ACL (1)*, pages 1074–1084. Association for Computational Linguistics.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 885–893. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18.

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 1684–1692.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

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.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2519–2531.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. *arXiv preprint arXiv:1906.00077*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of* *the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alfonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André FT Martins, and Shay B Cohen. 2019. Jointly extracting and compressing documents with summary state representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3955–3966.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299. International World Wide Web Conferences Steering Committee.

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. 2018a. Dont give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. 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*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 373–382. ACM.

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.

Xiaojun Wan, Ziqiang Cao, Furu Wei, Sujian Li, and Ming Zhou. 2015. Multi-document summarization via discriminative summary reranking. *arXiv preprint arXiv:1507.02062*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuan-Jing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Conference of the Association for Computational Linguistics*.

Danqing Wang, Pengfei Liu, Ming Zhong, Jie Fu, Xipeng Qiu, and Xuanjing Huang. 2019. Exploring domain shift in extractive text summarization. *arXiv preprint arXiv:1908.11664*.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144—4150. AAAI Press.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-aware neural extractive model for text summarization. *arXiv preprint arXiv:1910.14142*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753.

Dani Yogatama, Fei Liu, and Noah A Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966.

Haoyu Zhang, Yeyun Gong, Yu Yan, Nan Duan, Jianjun Xu, Ji Wang, Ming Gong, and Ming Zhou. 2019a. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019b. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *ACL*.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. 2019a. Searching for effective neural extractive summarization: What works and whats next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1049–1058.

Ming Zhong, Danqing Wang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2019b. A closer look at data bias in neural extractive summarization models. *EMNLP-IJCNLP 2019*, page 80.

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.