# Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network

**Ruipeng Jia**[1,2], **Yanan Cao**[1]*, **Hengzhu Tang**[1], **Fang Fang**[1]*, **Cong Cao**[1] and **Shi Wang**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]Institute of Computing Technology, Chinese Academy of Sciences
[1,2]{jiaruipeng, caoyanan, tanghengzhu, fangfang0703, caocong}@iie.ac.cn
[3]wangshi@ict.ac.cn

## Abstract

Sentence-level extractive text summarization is substantially a node classification task of network mining, adhering to the informative components and concise representations. There are lots of redundant phrases between extracted sentences, but it is difficult to model them exactly by the general supervised methods. Previous sentence encoders, especially BERT, specialize in modeling the relationship between source sentences. While, they have no ability to consider the overlaps of the target selected summary, and there are inherent dependencies among target labels of sentences. In this paper, we propose HAHSum (as shorthand for **H**ierarchical **A**ttentive **H**eterogeneous Graph for Text **Sum**marization), which well models different levels of information, including words and sentences, and spotlights redundancy dependencies between sentences. Our approach iteratively refines the sentence representations with redundancy-aware graph and delivers the label dependencies by message passing. Experiments on large scale benchmark corpus (CNN/DM, NYT, and NEWSROOM) demonstrate that HAHSum yields ground-breaking performance and outperforms previous extractive summarizers.

## 1 Introduction

Single document extractive summarization aims to select subset sentences and assemble them as informative and concise summaries. Recent advances (Nallapati et al., 2017; Zhou et al., 2018; Liu and Lapata, 2019; Zhong et al., 2020) focus on balancing the salience and redundancy of sentences, i.e. selecting the sentences with high semantic similarity to the gold summary and resolving the redundancy between selected sentences. Taking Table 1 for example, there are five sentences in a document, and each of them is assigned one salience score and one label indicating whether this sentence should be contained in the extracted summary. Although *sent1*, *sent3*, and *sent4* are assigned high salience score, just *sent3* and *sent4* are selected as the summary sentences (with label 1) because there are too much redundancy information between unselected *sent1* and selected *sent3*. That is to say, whether one sentence could be selected depends on its salience and the redundancy with other selected sentences. However, it is still difficult to model the dependency exactly.

| | Salience | Label | Sentence |
|---|---|---|---|
| *sent1*: | 0.7 | 0 | Deanna Holleran is charged in murder. |
| *sent2*: | 0.1 | 0 | Jackson County Prosecutor Jean Peters Baker announced today. |
| *sent3*: | 0.7 | 1 | Deanna Holleran faces a charge of traffic accident. |
| *sent4*: | 0.7 | 1 | The fatal traffic accident is a murder. |
| *sent5*: | 0.2 | 0 | It took the life of Marianna Hernandez near 9th Hardesty. |
| *Summary*: | | | Woman faces a charge of murder for a fatal traffic accident. |

Table 1: Simplified News from Jackson County Prosecutor. **Salience** score is an approximate estimation derived from semantic and **Label** is converted from gold summary to ensure the concision and accuracy of the extracted summaries.

Most of the previous approaches utilize autoregressive architecture (Narayan et al., 2018; Mendes et al., 2019; Liu and Lapata, 2019; Xu et al., 2020), which just models the unidirectional dependency between sentences, i.e., the state of the current sentence is based on previously sentence labels. These models are trained to predict the current sentence label given the ground truth labels of the previous sentences, while feeding the predicted labels of the previous sentences as input in inference phase. As we all know, the autoregressive paradigm faces error propagation and exposure bias problems (Ranzato et al., 2015). Besides, reinforcement learning is also introduced to consider the semantics of extracted summary (Narayan et al., 2018; Bae et al.,

*Corresponding authors: Yanan Cao and Fang Fang

2019), which combines the maximum-likelihood cross-entropy loss with the rewards from policy gradient to directly optimize the evaluation metric for the summarization task. Recently, the popular solution is to build a summarization system with two-stage decoder. These models extract salient sentences and then rewrite (Chen and Bansal, 2018; Bae et al., 2019), compress (Lebanoff et al., 2019; Xu and Durrett, 2019; Mendes et al., 2019), or match (Zhong et al., 2020) these sentences.

Previous models generally use top-*k* strategy as an optimal strategy: for different documents, the number of selected sentences is constant which conflicts with the real world. For example, almost all previous approaches extract three sentences from the source articles (top-3 strategy (Zhou et al., 2018; Liu and Lapata, 2019; Zhang et al., 2019b; Xu et al., 2020)), although 40% documents in CNN/DM contain more or less than 3-sentences oracle summary. That's because these approaches are difficult to measure the salience and redundancy simultaneously with error propagation. Notably, Mendes et al. (2019) introduces the length variable into the decoder and Zhong et al. (2020) can choose any number of sentences by match candidate summary in semantic space.

In order to address above issues, we construct the source article as a hierarchical heterogeneous graph (HHG) and propose a Graph Attention Net (Veličković et al., 2018) based model (HAHSum) to extract sentences by simultaneously balancing salience and redundancy. In HHG, both words and sentences are constructed as nodes, the relations between them are constructed as different types of edges. This hierarchical graph can be viewed as a two-level graph: word-level and sentence-level. For word-level graph (word-word), we design an **Abstract Layer** to learn the semantic representation of each word. Then, we transduce the word-level graph into the sentence-level one, by aggregating each word to its corresponding sentence node. For sentence-level graph (sentence-sentence), we design a **Redundancy Layer**, which firstly pre-labels each sentence and iteratively updates the label dependencies by propagating redundancy information. The redundancy layer restricts the scale of receptive field for redundancy information, and the information passing is guided by the ground-truth labels of sentences. After obtaining the redundancy-aware sentence representations, we use a classifier to label these sentence-level nodes with a threshold. In this way, the whole framework extracts summary sentences simultaneously instead of autoregressive paradigm, taking away the top-*k* strategy.

The contributions of this paper are as below: 1) We propose a hierarchical attentive heterogeneous graph based model(HAHSum) to guide the redundancy information propagating between sentences and learn redundancy-aware sentence representation; 2) Our architecture is able to extract flexible quantity of sentences with a threshold, instead of top-*k* strategy; 3) We evaluate HAHSum on three popular benchmarks (CNN/DM, NYT, NEWSROOM) and experimental results show that HAHSum outperforms the existing state-of-the-art approaches. Our source code will be available on Github [1].

## 2 Related Work

### 2.1 Extractive Summarization

Neural networks have achieved great success in the task of text summarization. There are two main lines of research: abstractive and extractive. The abstractive paradigm (Rush et al., 2015; See et al., 2017; Celikyilmaz et al., 2018; Sharma et al., 2019) focuses on generating a summary word-by-word after encoding the full document. The extractive approach (Cheng and Lapata, 2016; Zhou et al., 2018; Narayan et al., 2018) directly selects sentences from the document to assemble into a summary.

Recent research work on extractive summarization spans a large range of approaches. These work usually instantiate their encoder-decoder architecture by choosing RNN (Nallapati et al., 2017; Zhou et al., 2018), Transformer (Wang et al., 2019; Zhong et al., 2019b; Liu and Lapata, 2019; Zhang et al., 2019b) or Hierarchical GNN (Wang et al., 2020) as encoder, autoregressive (Jadhav and Rajan, 2018; Liu and Lapata, 2019) or non-autoregressive (Narayan et al., 2018; Arumae and Liu, 2018) decoders. The application of RL provides a means of summary-level scoring and brings improvement (Narayan et al., 2018; Bae et al., 2019).

### 2.2 Graph Neural Network for NLP

Recently, there is considerable amount of interest in applying GNN to NLP tasks and great success has been achieved. Fernandes et al. (2019) applied

[1]http://github.com/coder352/HAHSum

sequence GNN to model the sentences with named entity information. Yao et al. (2019) used two-layer GCN for text classification and introduced a well-designed adjacency matrix. GCN also played an important role in Chinese named entity (Ding et al., 2019). Liu et al. (2019) proposed a new contextualized neural network for sequence learning by leveraging various types of non-local contextual information in the form of information passing over GNN. These studies are related to our work in the sense that we explore extractive text summarization by message passing through hierarchical heterogeneous architecture.

## 3 Methodology

### 3.1 Problem Definition

Let $S = \{s_1, s_2, ..., s_N\}$ denotes the source document sequence which contains $N$ sentences, where $s_i$ is the $i$-th sentence of document. Let $T$ denotes the hand-crafted summary. Extractive summarization aims to produce summary $S^* = \{s_1^*, s_2^*, ..., s_M^*\}$ by selecting $M$ sentences from $S$, where $M \leq N$. Labels $Y = \{y_1, y_2, ..., y_N\}$ are derived from $T$, where $y_i \in \{0, 1\}$ denotes whether sentence $s_i$ should be included in the extracted summary. Oracle summary is a subset of $S$, which achieves the highest ROUGE score calculated with $T$.

### 3.2 Graph Construction

In order to model the redundancy relation between sentences, we use a heterogeneous graph which contains multi-granularity levels of information to represent a document, as shown in Figure 1. In this graph, there are three types of nodes: named entity, word, and sentence. To reduce semantic sparsity, we replace text spans of `Named Entity` by anonymized tokens (e.g. [Person_A], [Person_B], [Date_A]). `Word` node is the original textual item, representing word-level information. Different from DivGraphPointer (Sun et al., 2019), which aggregates identical words into one node, we keep each word occurrence as one node to avoid the confusion of different contexts. Each `Sentence` node corresponds to one sentence and represents the global information of one sentence.

We also define four types of edges to represent various structural information in HAHSum:

1. We connect sequential named entities and words in one sentence using directed `Next` edges.
2. We connect one named entity node or word node to one sentence node with directed `In` edge if the named entity or word occurs in this sentence.
3. We connect two named entity nodes with undirected `Same` edge if they are the same named entity.
4. We connect two sentence nodes with undirected `Similar` edge if they have trigram overlapping.

The topological structure of graph can be represented by adjacency matrix $A$, where the bool-type element is indicating whether there is an edge between nodes. Because HAHsum contains multi-granularity levels of information, it can be divided into three subgraphs: the word-level, word-sentence, and sentence-level subgraph. So, we define three adjacency matrices: $A_{word}$ is used for the word-level graph, constructed by `Entity` node, `Word` node, `Next` edge and `Same` edge. $A_{word-sent}$ is used for the word-sentence graph, constructed by three types of nodes and `In` edge. $A_{sent}$ is used for sentence-level graph, constructed by `Sentence` node and `Similar` edge. By propagating the information from word-level to sentence-level graph, we can obtain the sentence representation and model the redundancy between sentences.

Generally, the message passing over graphs can be achieved in two steps: aggregation and combination, and this process can be conducted multiple times (referred as layers or hops in GNN literature) (Tu et al., 2019). Therefore, we iteratively update the sentence nodes representation with redundancy message passing which will be described in the following sections.

### 3.3 Graph Attention Network

To represent graph structure $A$ and node content $X$ in a unified framework, we develop a variant of Graph Attention Network (GAT) (Veličković et al., 2018). GAT is used to learn hidden representations of each node by aggregating the information from its neighbors, with the attention coefficients:

$$e_{ij} = \mathrm{LeakReLU}(a(Wx_i||Wx_j)) \qquad (1)$$

where $W \in \mathbb{R}^{d \times d}$ is a shared linear transformation weight matrix for this layer, $||$ is the concatenation operation, and $a \in \mathbb{R}^{2d}$ is a shared attentional weight vector.

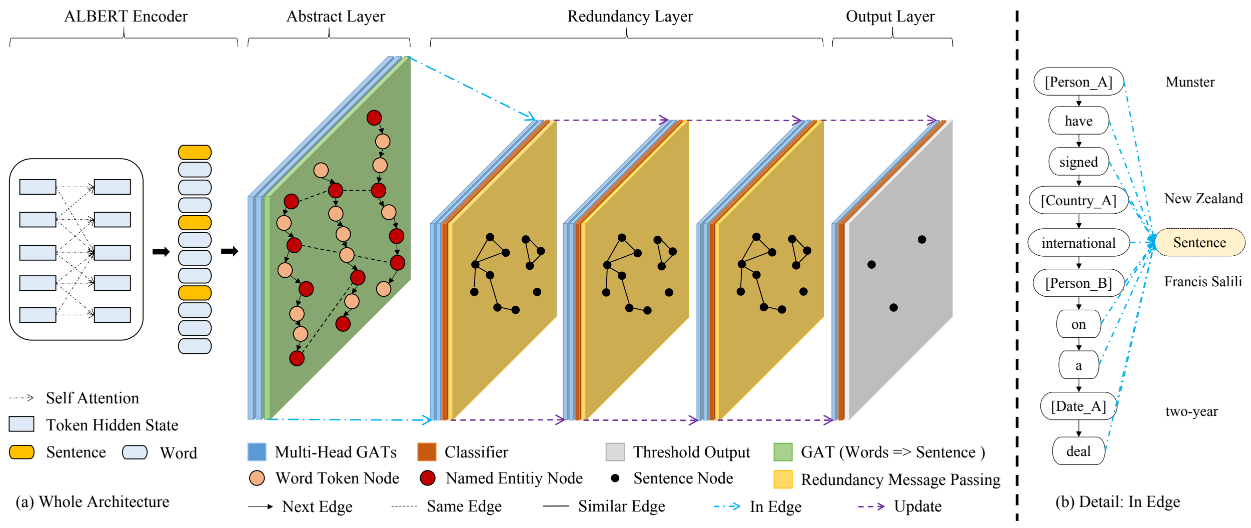


Figure 1: Overview of Hierarchical Attentive Heterogeneous Graph

To make the attention coefficients easily comparable across different nodes, we normalize them as follows:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \quad (2)$$

where $\mathcal{N}_i$ denotes the neighbors of node $i$ according to adjacency matrix $A$.

Then, the normalized attention coefficients are used to compute a linear combination of features.

$$x'_i = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W x_j) + W' x_i \quad (3)$$

where $W'$ is used to distinguish the information between $x_i$ and its neighbors.

## 3.4 Message Passing

Shown in Figure 1, HAHSum consists of ALBERT Encoder, Abstract Layer, Redundancy Layer, and Output Layer. We next introduce how the information propagates over these layers.

### 3.4.1 ALBERT Encoder

In order to learn the contextual representation of words, we use a pre-trained ALBERT (Lan et al., 2019) for summarization and the architecture is similar to BERTSUMEXT (Liu and Lapata, 2019). The output of ALBERT encoder contains word hidden states $h^{word}$ and sentence hidden states $h^{sent}$. Specifically, ALBERT takes subword units as input, which means that one word may correspond to multiple hidden states. In order to accurately use these hidden states to represent each word, we apply an average pooling function to the outputs of ALBERT.

### 3.4.2 Abstract Layer

The abstract layer contains three GAT sublayers which are described in Section 3.3: two for word-level graph and one for word-sentence transduction. The first two GAT sublayers are used to learn the hidden state of each word based on its two-order neighbors inspired by Kipf and Welling[2017],

$$\mathcal{W} = \text{GAT}(\text{GAT}(h^{word}, A_{word}), A_{word}) \quad (4)$$

where $A_{word}$ denotes the adjacency matrix of the word-level subgraph, and $\mathcal{W}$ denotes the hidden state of the word nodes.

The third GAT sublayer is to learn the initial representation of each sentence node, derived from the word hidden states:

$$[\mathcal{W}, \mathcal{S}_{abs}] = \text{GAT}([\mathcal{W}, h^{sent}], A_{word-sent}) \quad (5)$$

where $A_{word-sent}$ denotes the adjacency matrix of the word-sentence subgraphs, and $\mathcal{S}_{abs}$ (abs is for abstract) is the initial representation of sentence nodes.

### 3.4.3 Redundancy Layer

The BERT encoder and abstract layer specialize in modeling salience with overall context representation of sentences, while it is powerless for redundancy information with dependencies among target labels. So, redundancy layer aims to model the redundancy, by iteratively updating the sentence representation with redundancy message passing, and this process is supervised by ground-truth labels.

This layer only deals with sentence-level information $\mathcal{S} = \{h_1, h_2, ..., h_N\}$ and iteratively updates it $L$ times with classification scores:

$$\begin{aligned}\widetilde{\mathcal{S}}_{re}^{l} &= \mathrm{GAT}(\mathrm{GAT}(\mathcal{S}_{re}^{l}, A_{sent}), A_{sent}) \\ P(y_i = 1|\widetilde{\mathcal{S}}_{re}^{l}) &= \sigma(\mathrm{FFN}(\mathrm{LN}(\tilde{h}_i^l + \mathrm{MHAtt}(\tilde{h}_i^l))))\end{aligned} \quad (6)$$

where $\mathcal{S}_{re}^{0} = \mathcal{S}_{abs}$ (re is for redundancy) and we get $\mathcal{S}_{re}^{L}$ at the end, $W_c$, $W_r$ are weight parameters, FFN, LN, MHAtt are feed-foreard network, layer normalization and multi-head attention layer.

We update $\tilde{h}_i^l$ by reducing the redundancy information $g_i^l$, which is the weighted summation of neighbors information:

$$\begin{aligned} g_i^l &= \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} P(y_j = 1|\widetilde{\mathcal{S}}_{re}^{l}) * \tilde{h}_j^l \\ h_i^{l+1} &= W_c^l * \tilde{h}_i^l - \tilde{h}_i^{lT} W_r^l \tanh(g_i^l) \\ \mathcal{S}_{re}^{l+1} &= (h_1^{l+1}, h_2^{l+1}, ..., h_{|S|}^{l+1}) \end{aligned} \quad (7)$$

where $\mathcal{N}_i$ is redundancy receptive field for node $i$, according to $A_{sent}$.

Specifically, we employ a gating mechanism (Gilmer et al., 2017) for the information update, so that: 1) to avoid GNN smoothing problem; 2) the original overall information from ALBERT is accessible for the ultimate classifier.

$$\begin{aligned} \tilde{h}_i^{l'} &= W_c^l * \tilde{h}_i^l - \tilde{h}_i^{lT} W_r^l \tanh(g_i^l) \\ p_g^l &= \sigma(f_g^l([\tilde{h}_i^l; \tilde{h}_i^{l'}])) \\ h_i^{l+1} &= \tilde{h}_i^l \odot p_g^l + \tilde{h}_i^{l'} \odot (1 - p_g^l) \end{aligned} \quad (8)$$

where $\odot$ denotes element-wise multiplication.

## 3.5 Objective Function

Previous approaches for modeling the salience and redundancy is autoregressive, where observations from previous time-steps are used to predict the value at current time-step:

$$P(Y|S) = \prod_{t=1}^{|S|} P(y_t|S, y_1, y_2, ..., y_{t-1}) \quad (9)$$

The autoregressive models have some disadvantages: 1) the error in inference will propagate subsequently, 2) label $y_t$ is generated just depend on previous sentences $y_{<t}$ rather than considering bidirectional dependency, and 3) it is difficult to decide how many sentences to extract.

| Datasets | avg.doc length | | avg.summary length | |
|---|---|---|---|---|
| | words | sentences | words | sentences |
| CNN | 760.50 | 33.98 | 45.70 | 3.59 |
| DailyMail | 653.33 | 29.33 | 54.65 | 3.86 |
| NYT | 800.04 | 35.55 | 45.54 | 2.44 |
| Newsroom (Ext) | 605.44 | 28.78 | 40.95 | 1.90 |

Table 2: Data Statistics: CNN/Daily Mail, NYT, Newsroom

Our HAHSum predicts these labels simultaneously:

$$P(Y|S) = \prod_{t=1}^{|S|} P(y_t|S, \mathcal{S}_{abs}, \mathcal{S}_{re}) \quad (10)$$

where we extract flexible quantity of sentences with a threshold instead of top-$k$. For $L$ classifiers in our model, we train them simultaneously with different proportions. For each training pair $(X, Y)$ and the predicted $\hat{Y}$, the loss function is formalized as follows:

$$\begin{aligned}\mathcal{L} = -\sum_{l=0}^{L}\frac{L+l}{2L}\sum_{i=1}^{|S|}\{y_i \log P(\hat{y}_i|\widetilde{\mathcal{S}}_{re}^{l}) + \\ (1-y_i)\log(1 - P(\hat{y}_i|\widetilde{\mathcal{S}}_{re}^{l}))\}\end{aligned} \quad (11)$$

## 4 Experiments Setting

### 4.1 Benchmark Datasets

As shown in Table 2, we employ three datasets widely-used with multiple sentences summary (CNN/DM (Hermann et al., 2015), NYT (Sandhaus, 2008), and NEWSROOM (Grusky et al., 2018)). These summaries vary with respect to the type of rewriting operations, e.g., CNN/DM and NYT prefer to the abstractive approaches and Newsroom(Ext) is genuinely extractive. We employ the greedy method to obtain ground-truth sentence labels (Nallapati et al., 2017).

**CNN/DailyMail:** We use the standard splits for training, validation, and test (90,266/1,220/1,093 for CNN and 196,96/12,148/10,397 for DailyMail) (Liu and Lapata, 2019). Input documents are truncated to 768 BPE tokens, with anonymized entities and processed by Stanford CoreNLP.

**NYT:** Following previous work (Zhang et al., 2019b; Xu and Durrett, 2019), we use 137,778, 17,222 and 17,223 samples for training, validation, and test, respectively. Input documents were truncated to 768 BPE tokens too. Note that there are different division for NYT (Durrett et al., 2016; Liu and Lapata, 2019) and several models are not evaluated on NYT officially (See et al., 2017; Mendes et al., 2019), so we re-train and evaluate them on NYT with the source code from Github.

**Newsroom(Ext):** We employ the extractive part of Newsroom with same divisioin method (Mendes et al., 2019) for training/validation/test (331,778/36,332/36,122). Input documents are truncted to 768 BPE tokens.

### 4.2 Evaluation Metric & Parameter Settings

**Metric:** ROUGE (Lin, 2004) is the standard metric for evaluating the quality of summaries. We report the ROUGE-1, ROUGE-2, and ROUGE-L of HAHSum by *ROUGE-1.5.5.pl*, which calculates the overlap lexical units of extracted sentences and ground-truth.

**Graph Structure:** For abstract layer, we extract the named entities (`[Person]`, `[Date]`, `[Country]`, `[Buildings]`, `[Monetary]`) using CoreNLP, and replace them by anonymized tokens. Similar to Fernandes et al. (2019), we have tried to add dependency parse edges and they didn't show significant benefits, owing to the facts that 1) the dependency tree is substantially a permutation sequential structure, with little advancements for original information; 2) the performance is influenced by the accuracy of the upstream annotators. We have tried the iteration steps of $[1, 2, 3, 5]$ for updating redundancy layer, and $L = 3$ is the best value in experiment result.

**Parameters:** We employ pre-trained 'albert-xxlarge-v2'[2] and reuse the implementation of PreSumm[3]. We train our model (with about 400M parameters) one day for 100,000 steps on 2 GPUs(Nvidia Tesla V100, 32G) with gradient accumulation every two steps. We select the top-3 checkpoints according to the evaluation loss on validation set and report the averaged results on the test set. Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ is used as optimizer and learning rate schedule follows the strategies with warming-up on first 10,000 steps (Vaswani et al., 2017). The final threshold in extraction is 0.65 for CNN/DM, 0.58 for NYT and 0.64 for Newsroom, with the highest ROUGE-1 score individually. A higher threshold will be with more concise summary and the lower threshold will return more information.

[2]https://github.com/huggingface/transformers

[3]https://github.com/nlpyang/PreSumm

### 4.3 Baselines

**Extractive Methods:** **Oracle** is the extracted summary according to the ground-truth labels. **Lead** is a base method for extractive text summarization that chooses first several sentences as a summary. **SummaRuNNer** takes content, salience, novelty, and position of each sentence into consideration when deciding if a sentence should be included in the extractive summary. **PN-BERT** tries to employ the unsupervised transferable knowledge. **BERTSUMEXT** applies pre-trained BERT in text summarization and proposes a general framework for both extractive and abstractive models. **MATCHSUM** is a two-stage method for extract-then-match, and the first-stage is BERT-SUMEXT.

**Abstractive Methods:** **ABS** is the normal architecture with RNN-based encoder and decoder. **PGC** augments the standard Seq2Seq attentional model with pointer and coverage mechanisms. **TransformerABS** employs Transformer in text summarization. **MASS** proposes masked Seq2Seq pre-training for encoder-decoder. **UniLM** presents unified pre-trained language model, that can be fine-tuned for summarization. **BART**, and **Prophet-Net** are pre-trained on large unlabeled data and perform excellent performance with Transformer architecture. **PEGASUS** proposes Transformer-based models with extracted gap-sentences for abstractive summarization.

Specifically, these Transformer-based approaches are divided into `Base` and `Large` versions, according to the layers of Transformer.

## 5 Analysis

### 5.1 Rouge Scores

The experiment results on three benchmark datasets are shown in Table 3. There are ignored positions for Newsroom(Ext), which is designed for extractive approaches, eliminating the demanding of abstractive ones. It is obvious that HAHSum almost outperforms all the baselines across most of the evaluation metrics. For CNN/DM, there is little gap between the performance of extractive and abstractive architectures, particularly demonstrating the popularity and generality of this dataset. While NYT prefers to abstractive methods, and NEWSROOM(Ext) is constructed by extracting sentences

| Models | CNN/DM | | | NYT | | | Newsroom (Ext) | | |
|---|---|---|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| **Abstractive** | | | | | | | | | |
| ABS (2015) | 35.46 | 13.30 | 32.65 | 42.78 | 25.61 | 35.26 | 6.1 | 0.2 | 5.4 |
| PGC (2017) | 39.53 | 17.28 | 36.38 | 43.93 | 26.85 | 38.67 | 39.1 | 27.9 | 36.2 |
| TransformerABS (2017) | 40.21 | 17.76 | 37.09 | 45.36 | 27.34 | 39.53 | 40.3 | 28.7 | 36.5 |
| MASS$_{\text{Large}}$ (2019) | 43.05 | 20.02 | 40.08 | - | - | - | - | - | - |
| UniLM$_{\text{Large}}$ (2019) | 43.33 | 20.21 | 40.51 | - | - | - | - | - | - |
| BART$_{\text{Large}}$ (2019) | 44.16 | 21.28 | 40.90 | 48.73 | 29.25 | 44.48 | - | - | - |
| PEGASUS$_{\text{Large}}$ (2019a) | 44.17 | 21.47 | 41.11 | - | - | - | - | - | - |
| ProphetNet$_{\text{Large}}$ (2020) | 44.20 | 21.17 | 41.30 | - | - | - | - | - | - |
| **Extractive** | | | | | | | | | |
| Oracle | 55.61 | 32.84 | 51.88 | 64.22 | 44.57 | 57.27 | - | - | - |
| Lead | 40.42 | 17.62 | 36.67 | 41.80 | 22.60 | 35.00 | 53.1 | 49.0 | 52.4 |
| SummaRuNNer (2017) | 39.60 | 16.20 | 35.30 | 42.37 | 23.89 | 38.74 | 48.96 | 44.33 | 49.57 |
| Exconsumm (2019) | 41.7 | 18.6 | 37.8 | 43.18 | 24.43 | 38.92 | 68.4 | 62.9 | 67.3 |
| PNBERT$_{\text{Base}}$ (2019a) | 42.69 | 19.60 | 38.85 | - | - | - | - | - | - |
| BERTSUMEXT$_{\text{Large}}$ (2019) | 43.85 | 20.34 | 39.90 | 48.51 | 30.27 | 44.65 | 70.85 | 67.03 | 69.61 |
| MATCHSUM$_{\text{Base}}$ (2020) | 44.41 | 20.86 | 40.55 | - | - | - | - | - | - |
| **HAHSum**$_{\text{Large}}$(Ours) | **44.68** | **21.30** | **40.75** | **49.36** | **31.41** | **44.97** | **71.31** | **68.75** | **70.83** |

Table 3: Automatic Evaluation or ROUGE

directly. For extractive approaches, HAHSum, MATCHSUM, and BERTSUMEXT are outstanding with the power of pre-trained BERT-like models. For abstractive methods, these variants of Transformer perform extremely with deep architectures and large-scale unlabeled corpus.

HAHSum outperforms all other extractive approaches for that: 1) HAHSum achieves improvements to mitigate the redundancy bias by measuring salience and redundancy simultaneously, while this would not be possible with any framework in the autoregressive literature because salience and redundancy are treated as two different processes due to the dependency among target labels. 2) The promising results of heterogeneous sequence-graph models outperform pure sequence models. Sequence encoders with a graph component can reason about long-distance relationships in weakly structured data such as text, which requires non-trivial understanding of the input, while attentive sequential architectures prefer to calculate the relevance merely.

## 5.2 Ablation Studies

We propose several strategies to improve the performance by relieving the semantic sparsity and redundancy bias, including abstract layer(AL), the iterative redundancy layer(RL), and pre-trained ALBERT. To investigate the influence of these factors, we conduct the experiments and list the results in Table 4. Significantly, AL is more important than RL, for the reason that there are lots of meaningless named entities. Besides, RL mechanism enlarges the advantage of extraction without top-*k* strategy, for there are more than 40% documents in CNN/DM contains more or less than 3-sentences oracle summary. As shown in Table 6, HAHSum predicts sequence exactly with two sentences, same as the oracle summary. While BERTSUMEXT extracts top-3 sentences strictly, in spite of the inaccurateness and redundancy.

| Models | R-1 | R-2 | R-L |
|---|---|---|---|
| HAHSum | **44.68** | **21.30** | **40.75** |
| w/o AL | 44.35 | 20.98 | 40.49 |
| w/o RL | 44.49 | 21.11 | 40.58 |
| w/o ALBERT | 44.57 | 21.14 | 40.53 |

Table 4: Ablation Study on CNN/DM Test Set

## 5.3 Human Evaluation for Summarization

It is not enough only relying on the ROUGE evaluation for a summarization system, although the ROUGE correlates well with human judgments (Owczarzak et al., 2012). To evaluate the performance of HAHSum more accurately, we design

| Models | 1st | 2nd | 3rd | 4th | 5th | MeanR |
|---|---|---|---|---|---|---|
| SummaRuNNer | 0.14 | 0.27 | 0.24 | 0.22 | 0.13 | 2.93 |
| BERTSUMEXT | 0.20 | 0.28 | 0.31 | 0.16 | 0.05 | 2.58 |
| MATCHSUM | 0.24 | 0.36 | 0.16 | 0.15 | 0.09 | 2.49 |
| HAHSum | 0.45 | 0.34 | 0.18 | 0.03 | 0.00 | 2.24 |
| Ground-Truth | 0.70 | 0.21 | 0.05 | 0.04 | 0.00 | 1.43 |

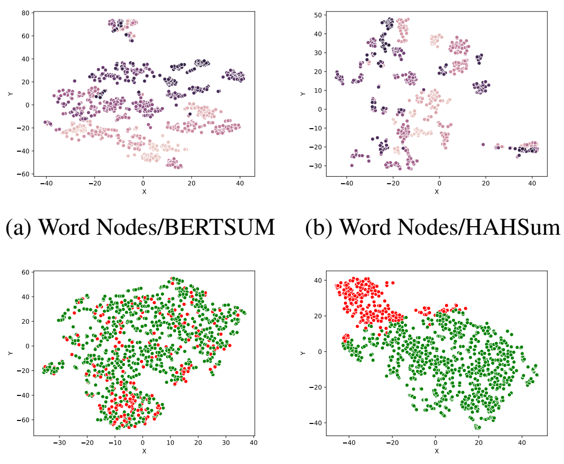
Table 5: Human evaluation on Daily Mail.

an Amazon Mechanical Turk experiment based on ranking method. Following Cheng and Lapata (2016); Narayan et al. (2018); Zhang et al. (2019b), firstly, we randomly select 40 samples from Daily Mail test set. Then the human participants are presented with a original document and a list of corresponding summaries produced by different model systems. Participants are requested to rank these summaries (ties allowed) by taking informativeness (Can the summary capture the important information from the document) and fluency (Is the summary grammatical) into account. Each document is annotated by three different participants separately.

Following the previous work, the input article and ground truth summaries are also shown to the human participants in addition to the four model summaries (SummaRuNNer, BERTSUMEXT, MATCHSUM and HAHSum). From the results shown in Table 5, we can see that HAHSum is better in relevance compared with others.

## 5.4 Visualization

We visualize the learned embedding of word and sentence nodes in a two-dimensional space by applying the t-SNE algorithm. We randomly select 500 continuous word nodes (approximately 30 sentences in a document) and 1000 sentence nodes from BERTSUMEXT and HAHSum separately. As shown in Figure 2, for word nodes, the darkness determines it's position in one document; while for sentence nodes, red points are the sentences with label 1, and green points are with label 0. The result shows: 1) It is amazing that sentence-level summarization constrains word representations to be shared across whole sentence, and there are obviously word clusters in BERTSUMEXT; 2) The word clusters are more distinct and meaningful in HAHSum equipped with abstract layer and GAT; 3) Intuitively, the redundancy layer has particularly strong representation power and generalizability, for that oracle sentence nodes in HAHSum are easy to identify, without autoregressive formalism used for capturing sentence-level redundancy.


(a) Word Nodes/BERTSUM (b) Word Nodes/HAHSum

(c) Sent Nodes/BERTSUM (d) Sent Nodes/HAHSum

Figure 2: T-SNE Visualization on CNN/DM Test Set

| |
|---|
| **Source Document (truncated):** built at a cost of # 1 billion, new broadcasting house is the jewel in the crown of the bbc and the setting for its self-mocking satire w1a. (...) new broadcasting house is home to three 24-hour news channels, nine radio networks and 6,000 staff. |
| **Oracle Summary:** new broadcasting house was opened by the queen in 2013–four years behind schedule and at least #55 million over budget. the bbc has admitted it 'occasionally' runs out of meeting rooms in its # 1billion new broadcasting house |
| **HAHSum:** new broadcasting house was opened by the queen in 2013–four years behind schedule and at least #55 million over budget. the bbc has admitted it 'occasionally' runs out of meeting rooms in its # 1billion new broadcasting house |
| **BERTSUMEXT:** new broadcasting house was opened by the queen in 2013–four years behind schedule and at least #55 million over budget. another said: 'it's bonkers to hold meetings across the street.' a bbc spokesman said: 'it is occasionally necessary to book nearby venues, especially for larger meetings. |

Table 6: Case Study on CNN/DM Test Set

## 6 Conclusion

In this paper, we propose hierarchical attentive heterogeneous graph, aiming to advance text summarization by measuring salience and redundancy simultaneously. Our approach model redundancy information by iteratively update the sentence information with message passing in redundancy-aware graph. As a result, HAHSum produces more focused summaries with fewer superfluous and the performance improvements are more pronounced on more extractive datasets.

## Acknowledgments

This research is supported by the National Key Research and Development Program of China (NO.2018YFB1004703). We thank all authors for their contributions and all anonymous reviewers for their constructive comments.

## References

Kristjan Arumae and Fei Liu. 2018. Reinforced extractive summarization with question-focused rewards. In *ACL*.

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In *Conference on Empirical Methods in Natural Language Processing, Workshop on New Frontiers in Summarization*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *NAACL-HLT*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*.

Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A neural multi-digraph model for chinese ner with gazetteers. In *ACL*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NIPS*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *ACL*.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Structured neural summarization. In *ICLR*.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *ACL*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

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. *ACL*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2019. Contextualized non-local neural networks for sequence learning. In *AAAI*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP*.

Afonso 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 *NAACL-HLT*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. In *ICLR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Evan Sandhaus. 2008. The new york times annotated corpus. In *Linguistic Data Consortium, Philadelphia*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. In *EMNLP*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*.

Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *ACL*.

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 *NIPS*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *ACL*.

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

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. *EMNLP*.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *ACL*.

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *arXiv preprint arXiv:2001.04063*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *AAAI*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *arXiv preprint arXiv:1912.08777*.

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, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *ACL*.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019a. Searching for effective neural extractive summarization: What works and whats next. In *ACL*.

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

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 *ACL*.