# HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization

**Xingxing Zhang, Furu Wei** and **Ming Zhou**
Microsoft Research Asia, Beijing, China
{xizhang,fuwei,mingzhou}@microsoft.com

## Abstract

Neural extractive summarization models usually employ a hierarchical encoder for document encoding and they are trained using sentence-level labels, which are created heuristically using rule-based methods. Training the hierarchical encoder with these *inaccurate* labels is challenging. Inspired by the recent work on pre-training transformer sentence encoders (Devlin et al., 2018), we propose HIBERT (as shorthand for **HI**erachical **B**idirectional **E**ncoder **R**epresentations from **T**ransformers) for document encoding and a method to pre-train it using unlabeled data. We apply the pre-trained HIBERT to our summarization model and it outperforms its randomly initialized counterpart by 1.25 ROUGE on the CNN/Dailymail dataset and by 2.0 ROUGE on a version of New York Times dataset. We also achieve the state-of-the-art performance on these two datasets.

## 1 Introduction

Automatic document summarization is the task of rewriting a document into its shorter form while still retaining its important content. Over the years, many paradigms for document summarization have been explored (see Nenkova and McKeown (2011) for an overview). The most popular two among them are *extractive* approaches and *abstractive* approaches. As the name implies, *extractive* approaches generate summaries by *extracting* parts of the original document (usually sentences), while *abstractive* methods may generate new words or phrases which are not in the original document.

Extractive summarization is usually modeled as a sentence ranking problem with length constraints (e.g., max number of words or sentences). Top ranked sentences (under constraints) are selected as summaries. Early attempts mostly leverage manually engineered features (Filatova and

Hatzivassiloglou, 2004a). Based on these sparse features, sentence are selected using a classifier or a regression model. Later, the feature engineering part in this paradigm is replaced with neural networks. Cheng and Lapata (2016) propose a hierarchical long short-term memory network (LSTM; Hochreiter and Schmidhuber 1997) to encode a document and then use another LSTM to predict binary labels for each sentence in the document. This architecture is widely adopted recently (Nallapati et al., 2017; Narayan et al., 2018; Zhang et al., 2018). Our model also employs a hierarchical document encoder, but we adopt a hierarchical transformer (Vaswani et al., 2017) rather a hierarchical LSTM. Because recent studies (Vaswani et al., 2017; Devlin et al., 2018) show the transformer model performs better than LSTM in many tasks.

Abstractive models do not attract much attention until recently. They are mostly based on sequence to sequence (seq2seq) models (Bahdanau et al., 2015), where a document is viewed a sequence and its summary is viewed as another sequence. Although seq2seq based summarizers can be equipped with copy mechanism (Gu et al., 2016; See et al., 2017), coverage model (See et al., 2017) and reinforcement learning (Paulus et al., 2017), there is still no guarantee that the generated summaries are grammatical and convey the same meaning as the original document does. It seems that extractive models are more reliable than their abstractive counterparts.

However, extractive models require sentence level labels, which are usually not included in most summarization datasets (most datasets only contain document-summary pairs). Sentence labels are usually obtained by rule-based methods (e.g., maximizing the ROUGE score between a set of sentences and reference summaries) and may not be accurate. Extractive models proposed re-

cently (Cheng and Lapata, 2016; Nallapati et al., 2017) employ hierarchical document encoders and even have neural decoders, which are complex. Training such complex neural models with *inaccurate binary* labels is challenging. We observed in our initial experiments on one of our dataset that our extractive model (see Section 3.3 for details) overfits to the training set quickly after the second epoch, which indicates the training set may not be fully utilized. Inspired by the recent pre-training work in natural language processing (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018), our solution to this problem is to first pre-train the "complex"' part (i.e., the hierarchical encoder) of the extractive model on unlabeled data and then we learn to classify sentences with our model initialized from the pre-trained encoder. In this paper, we propose HIBERT, which stands for **HI**erachical **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. We design an unsupervised method to pre-train HIBERT for document modeling. We apply the pre-trained HIBERT to the task of document summarization and achieve state-of-the-art performance on both the CNN/Dailymail and New York Times dataset.

## 2   Related Work

In this section, we introduce work on extractive summarization, abstractive summarization and pre-trained natural language processing models. For a more comprehensive review of summarization, we refer the interested readers to Nenkova and McKeown (2011) and Mani (2001).

**Extractive Summarization**   Extractive summarization aims to select important sentences (sometimes other textual units such as elementary discourse units (EDUs)) from a document as its summary. It is usually modeled as a sentence ranking problem by using the scores from classifiers (Kupiec et al., 1995), sequential labeling models (Conroy and O'leary, 2001) as well as integer linear programmers (Woodsend and Lapata, 2010). Early work with these models above mostly leverage human engineered features such as sentence position and length (Radev et al., 2004), word frequency (Nenkova et al., 2006) and event features (Filatova and Hatzivassiloglou, 2004b).

As the very successful applications of neural networks to a wide range of NLP tasks, the manually engineered features (for document encoding) are replaced with hierarchical LSTMs/CNNs

and the sequence labeling (or classification) model is replaced with an LSTM decoder (Cheng and Lapata, 2016; Nallapati et al., 2017). The architecture is widely adopted in recent neural extractive models and is extended with reinforcement learning (Narayan et al., 2018; Dong et al., 2018), latent variable models (Zhang et al., 2018), joint scoring (Zhou et al., 2018) and iterative document representation (Chen et al., 2018). Recently, transformer networks (Vaswani et al., 2017) achieves good performance in machine translation (Vaswani et al., 2017) and a range of NLP tasks (Devlin et al., 2018; Radford et al., 2018). Different from the extractive models above, we adopt a hierarchical Transformer for document encoding and also propose a method to pre-train the document encoder.

**Abstractive Summarization**   Abstractive summarization aims to generate the summary of a document with rewriting. Most recent abstractive models (Nallapati et al., 2016) are based on neural sequence to sequence learning (Bahdanau et al., 2015; Sutskever et al., 2014). However, the generated summaries of these models can not be controlled (i.e., their meanings can be quite different from the original and contents can be repeated). Therefore, copy mechanism (Gu et al., 2016), coverage model (See et al., 2017) and reinforcement learning model optimizing ROUGE (Paulus et al., 2017) are introduced. These problems are alleviated but not solved. There is also an interesting line of work combining extractive and abstractive summarization with reinforcement learning (Chen and Bansal, 2018), fused attention (Hsu et al., 2018) and bottom-up attention (Gehrmann et al., 2018). Our model, which is a very good extractive model, can be used as the sentence extraction component in these models and potentially improves their performance.

**Pre-trained NLP Models**   Most model pre-training methods in NLP leverage the natural ordering of text. For example, word2vec uses the surrounding words within a fixed size window to predict the word in the middle with a log bilinear model. The resulting word embedding table can be used in other downstream tasks. There are other word embedding pre-training methods using similar techniques (Pennington et al., 2014; Bojanowski et al., 2017). Peters et al. (2018) and Radford et al. (2018) find even a sentence encoder
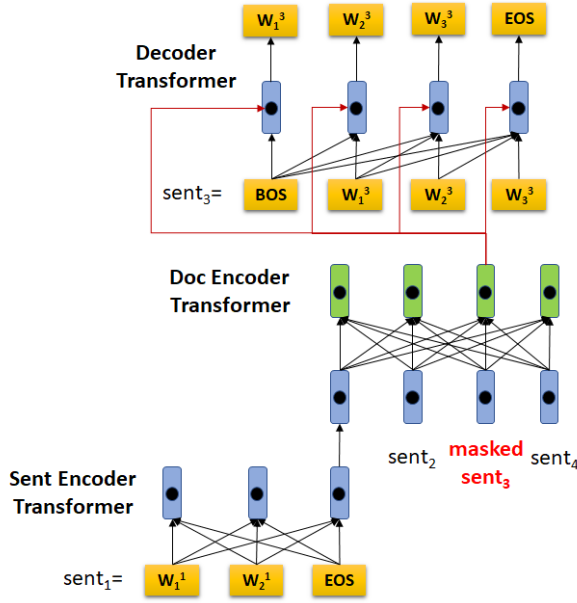
Figure 1: The architecture of HIBERT during training. $\text{sent}_i$ is a sentence in the document above, which has four sentences in total. $\text{sent}_3$ is masked during encoding and the decoder predicts the original $\text{sent}_3$.

(not just word embeddings) can also be pre-trained with language model objectives (i.e., predicting the next or previous word). Language model objective is unidirectional, while many tasks can leverage the context in both directions. Therefore, Devlin et al. (2018) propose the naturally bidirectional masked language model objective (i.e., masking several words with a special token in a sentence and then predicting them). All the methods above aim to pre-train word embeddings or sentence encoders, while our method aims to pre-train the hierarchical document encoders (i.e., hierarchical transformers), which is important in summarization.

## 3 Model

In this section, we present our model HIBERT. We first introduce how documents are represented in HIBERT. We then describe our method to pre-train HIBERT and finally move on to the application of HIBERT to summarization.

### 3.1 Document Representation

Let $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ denote a document, where $S_i = (w_1^i, w_2^i, \ldots, w_{|S_i|}^i)$ is a sentence in $\mathcal{D}$ and $w_j^i$ a word in $S_i$. Note that following common practice in natural language processing literatures, $w_{|S_i|}^i$ is an artificial EOS (End Of Sentence) token.

To obtain the representation of $\mathcal{D}$, we use two encoders: a *sentence encoder* to transform each sentence in $\mathcal{D}$ to a vector and a *document encoder* to learn sentence representations given their surrounding sentences as context. Both the *sentence encoder* and *document encoder* are based on the Transformer encoder described in Vaswani et al. (2017). As shown in Figure 1, they are nested in a hierarchical fashion. A transformer encoder usually has multiple layers and each layer is composed of a multi-head self attentive sub-layer followed by a feed-forward sub-layer with residual connections (He et al., 2016) and layer normalizations (Ba et al., 2016). For more details of the Transformer encoder, we refer the interested readers to Vaswani et al. (2017). To learn the representation of $S_i$, $S_i = (w_1^i, w_2^i, \ldots, w_{|S_i|}^i)$ is first mapped into continuous space

$$\mathbf{E}_i = (\mathbf{e}_1^i, \mathbf{e}_2^i, \ldots, \mathbf{e}_{|S_i|}^i)$$
$$\text{where} \quad \mathbf{e}_j^i = e(w_j^i) + \mathbf{p}_j \tag{1}$$

where $e(w_j^i)$ and $\mathbf{p}_j$ are the word and positional embeddings of $w_j^i$, respectively. The word embedding matrix is randomly initialized and we adopt the sine-cosine positional embedding (Vaswani et al., 2017)[1]. Then the *sentence encoder* (a Transformer) transforms $\mathbf{E}_i$ into a list of hidden representations $(\mathbf{h}_1^i, \mathbf{h}_2^i, \ldots, \mathbf{h}_{|S_i|}^i)$. We take the last hidden representation $\mathbf{h}_{|S_i|}^i$ (i.e., the representation at the EOS token) as the representation of sentence $S_i$. Similar to the representation of each word in $S_i$, we also take the sentence position into account. The final representation of $S_i$ is

$$\hat{\mathbf{h}}_i = \mathbf{h}_{|S_i|}^i + \mathbf{p}_i \tag{2}$$

Note that words and sentences share the same positional embedding matrix.

In analogy to the *sentence encoder*, as shown in Figure 1, the *document encoder* is yet another Transformer but applies on the sentence level. After running the Transformer on a sequence of sentence representations $(\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_{|\mathcal{D}|})$, we obtain the context sensitive sentence representations $(\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{|\mathcal{D}|})$. Now we have finished the encoding of a document with a hierarchical bidirectional transformer encoder HIBERT. Note that in previous work, document representation are also

---

[1] We use the sine-cosine embedding because it works well and do not introduce additional trainable parameters.

learned with hierarchical models, but each hierarchy is a Recurrent Neural Network (Nallapati et al., 2017; Zhou et al., 2018) or Convolutional Neural Network (Cheng and Lapata, 2016). We choose the Transformer because it outperforms CNN and RNN in machine translation (Vaswani et al., 2017), semantic role labeling (Strubell et al., 2018) and other NLP tasks (Devlin et al., 2018). In the next section we will introduce how we train HIBERT with an unsupervised training objective.

## 3.2 Pre-training

Most recent encoding neural models used in NLP (e.g., RNNs, CNNs or Transformers) can be pre-trained by predicting a word in a sentence (or a text span) using other words within the same sentence (or span). For example, ELMo (Peters et al., 2018) and OpenAI-GPT (Radford et al., 2018) predict a word using all words on its left (or right); while word2vec (Mikolov et al., 2013) predicts one word with its surrounding words in a fixed window and BERT (Devlin et al., 2018) predicts (masked) missing words in a sentence given all the other words.

All the models above learn the representation of a sentence, where its basic units are words. HIBERT aims to learn the representation of a document, where its basic units are sentences. Therefore, a natural way of pre-training a document level model (e.g., HIBERT) is to predict a sentence (or sentences) instead of a word (or words). We could predict a sentence in a document with all the sentences on its left (or right) as in a (document level) language model. However, in summarization, context on both directions are available. We therefore opt to predict a sentence using all sentences on both its left and right.

**Document Masking** Specifically, suppose $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ is a document, where $S_i = (w_1^i, w_2^i, \ldots, w_{|S_i|}^i)$ is a sentence in it. We randomly select 15% of the sentences in $\mathcal{D}$ and mask them. Then, we predict these masked sentences. The prediction task here is similar with the *Cloze* task (Taylor, 1953; Devlin et al., 2018), but the missing part is a sentence. However, during test time the input document is not masked, to make our model can adapt to documents without masks, we do not always mask the selected sentences. Once a sentence is selected (as one of the 15% selected masked sentences), we transform it with one of three methods below. We will use an ex-

ample to demonstrate the transformation. For instance, we have the following document and the second sentence is selected[2]:

```
William Shakespeare is a poet .
He died in 1616 .  He is regarded
as the greatest writer .
```

In 80% of the cases, we mask the selected sentence (i.e., we replace each word in the sentence with a mask token [MASK]). The document above becomes `William Shakespeare is a poet .  [MASK] [MASK] [MASK] [MASK] [MASK] He is regarded as the greatest writer .` (where "`He died in 1616 .  `" is masked).

In 10% of the cases, we keep the selected sentence as it is. This strategy is to simulate the input document during test time (with no masked sentences).

In the rest 10% cases, we replace the selected sentence with a random sentence. In this case, the document after transformation is `William Shakespeare is a poet .  Birds can fly .  He is regarded as the greatest writer .` The second sentence is replaced with "`Birds can fly .`" This strategy intends to add some noise during training and make the model more robust.

**Sentence Prediction** After the application of the above procedures to a document $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$, we obtain the masked document $\widetilde{\mathcal{D}} = (\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_{|\mathcal{D}|})$. Let $\mathcal{K}$ denote the set of indicies of selected sentences in $\mathcal{D}$. Now we are ready to predict the masked sentences $\mathcal{M} = \{S_k | k \in \mathcal{K}\}$ using $\widetilde{\mathcal{D}}$. We first apply the hierarchical encoder HIBERT in Section 3.1 to $\widetilde{\mathcal{D}}$ and obtain its context sensitive sentence representations $(\tilde{\mathbf{d}}_1, \tilde{\mathbf{d}}_2, \ldots, \tilde{\mathbf{d}}_{|\mathcal{D}|})$. We will demonstrate how we predict the masked sentence $S_k = (w_0^k, w_1^k, w_2^k, \ldots, w_{|S_k|}^k)$ one word per step ($w_0^k$ is an artificially added BOS token). At the $j$th step, we predict $w_j^k$ given $w_0^k, \ldots, w_{j-1}^k$ and $\widetilde{\mathcal{D}}$. $\tilde{\mathbf{d}}_k$ already encodes the information of $\widetilde{\mathcal{D}}$ with a focus around its $k$th sentence $\tilde{S}_k$. As shown in Figure 1, we employ a Transformer decoder (Vaswani et al., 2017) to predict $w_j^k$ with $\tilde{\mathbf{d}}_k$ as its additional input. The transformer decoder we used here is slightly different from the original one. The original decoder employs two multi-head attention layers to

---

[2]There might be multiple sentences selected in a document, but in this example there is only one.

include both the context in encoder and decoder, while we only need one to learn the decoder context, since the context in encoder is a vector (i.e., $\tilde{\mathbf{d}}_k$). Specifically, after applying the word and positional embeddings to $(w_0^k, \ldots, w_{j-1}^k)$, we obtain $\widetilde{\mathbf{E}}_{1:j-1}^k = (\tilde{\mathbf{e}}_0^k, \ldots, \tilde{\mathbf{e}}_{j-1}^k)$ (also see Equation 1). Then we apply multi-head attention sub-layer to $\widetilde{\mathbf{E}}_{1:j-1}^k$:

$$
\begin{aligned}
\tilde{\mathbf{h}}_{j-1} &= \text{MultiHead}(\mathbf{q}_{j-1}, \mathbf{K}_{j-1}, \mathbf{V}_{j-1}) \\
\mathbf{q}_{j-1} &= \mathbf{W}^Q \, \tilde{\mathbf{e}}_{j-1}^k \\
\mathbf{K}_{j-1} &= \mathbf{W}^K \, \widetilde{\mathbf{E}}_{1:j-1}^k \\
\mathbf{K}_{j-1} &= \mathbf{W}^V \, \widetilde{\mathbf{E}}_{1:j-1}^k
\end{aligned}
\tag{3}
$$

where $\mathbf{q}_{j-1}$, $\mathbf{K}_{j-1}$, $\mathbf{V}_{j-1}$ are the input query, key and value matrices of the multi-head attention function (Vaswani et al., 2017) MultiHead$(\cdot, \cdot, \cdot)$, respectively. $\mathbf{W}^Q \in \mathbb{R}^{d \times d}$, $\mathbf{W}^K \in \mathbb{R}^{d \times d}$ and $\mathbf{W}^V \in \mathbb{R}^{d \times d}$ are weight matrices.

Then we include the information of $\widetilde{\mathcal{D}}$ by addition:

$$
\tilde{\mathbf{x}}_{j-1} = \tilde{\mathbf{h}}_{j-1} + \tilde{\mathbf{d}}_k \tag{4}
$$

We also follow a feedforward sub-layer (one hidden layer with ReLU (Glorot et al., 2011) activation function) after $\tilde{\mathbf{x}}_{j-1}$ as in Vaswani et al. (2017):

$$
\tilde{\mathbf{g}}_{j-1} = \mathbf{W}_2^{ff} \max(0, \mathbf{W}_1^{ff} \tilde{\mathbf{x}}_{j-1} + \mathbf{b}_1) + \mathbf{b}_2 \tag{5}
$$

Note that the transformer decoder can have multiple layers by applying Equation (3) to (5) multiple times and we only show the computation of one layer for simplicity.

The probability of $w_j^k$ given $w_0^k, \ldots, w_{j-1}^k$ and $\widetilde{\mathcal{D}}$ is:

$$
p(w_j^k|w_{0:j-1}^k, \widetilde{\mathcal{D}}) = \text{softmax}(\mathbf{W}^O \, \tilde{\mathbf{g}}_{j-1}) \tag{6}
$$

Finally the probability of all masked sentences $\mathcal{M}$ given $\widetilde{\mathcal{D}}$ is

$$
p(\mathcal{M}|\widetilde{\mathcal{D}}) = \prod_{k \in \mathcal{K}} \prod_{j=1}^{|S_k|} p(w_j^k|w_{0:j-1}^k, \widetilde{\mathcal{D}}) \tag{7}
$$

The model above can be trained by minimizing the negative log-likelihood of all masked sentences given their paired documents. We can in theory have unlimited amount of training data for HIBERT, since they can be generated automatically from (unlabeled) documents. Therefore, we can first train HIBERT on large amount of data and then apply it to downstream tasks. In the next section, we will introduce its application to document summarization.
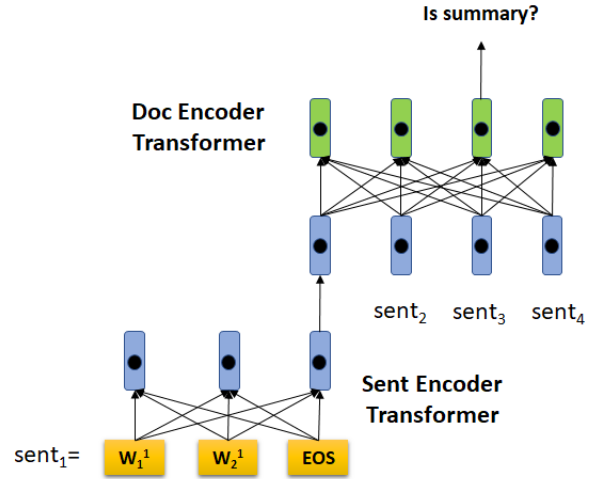


Figure 2: The architecture of our extractive summarization model. The sentence and document level transformers can be pretrained.

## 3.3 Extractive Summarization

Extractive summarization selects the most important sentences in a document as its summary. In this section, summarization is modeled as a sequence labeling problem. Specifically, a document is viewed as a sequence of sentences and a summarization model is expected to assign a `True` or `False` label for each sentence, where `True` means this sentence should be included in the summary. In the following, we will introduce the details of our summarization model based HIBERT.

Let $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ denote a document and $Y = (y_1, y_2, \ldots, y_{|\mathcal{D}|})$ its sentence labels (methods for obtaining these labels are in Section 4.1). As shown in Figure 2, we first apply the hierarchical bidirectional transformer encoder HIBERT to $\mathcal{D}$ and yields the context dependent representations for all sentences $(\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{|\mathcal{D}|})$. The probability of the label of $S_i$ can be estimated using an additional linear projection and a softmax:

$$
p(y_i|\mathcal{D}) = \text{softmax}(\mathbf{W}^S \, \mathbf{d}_i) \tag{8}
$$

where $\mathbf{W}^S \in \mathbb{R}^{2 \times d}$. The summarization model can be trained by minimizing the negative log-likelihood of all sentence labels given their paired documents.

## 4 Experiments

In this section we assess the performance of our model on the document summarization task. We

first introduce the dataset we used for pre-training and the summarization task and give implementation details of our model. We also compare our model against multiple previous models.

## 4.1 Datasets

We conducted our summarization experiments on the non-anonymous version CNN/Dailymail (CNNDM) dataset (Hermann et al., 2015; See et al., 2017), and the New York Times dataset (Durrett et al., 2016; Xu and Durrett, 2019). For the CNNDM dataset, we preprocessed the dataset using the scripts from the authors of See et al. (2017)[3]. The resulting dataset contains 287,226 documents with summaries for training, 13,368 for validation and 11,490 for test. Following (Xu and Durrett, 2019; Durrett et al., 2016), we created the NYT50 dataset by removing the documents whose summaries are shorter than 50 words from New York Times dataset. We used the same training/validation/test splits as in Xu and Durrett (2019), which contain 137,778 documents for training, 17,222 for validation and 17,223 for test. To create sentence level labels for extractive summarization, we used a strategy similar to Nallapati et al. (2017). We label the subset of sentences in a document that maximizes ROUGE (Lin, 2004) (against the human summary) as `True` and all other sentences as `False`.

To unsupervisedly pre-train our document model HIBERT (see Section 3.2 for details), we created the GIGA-CM dataset (totally 6,626,842 documents and 2,854 million words), which includes 6,339,616 documents sampled from the English Gigaword[4] dataset and the training split of the CNNDM dataset. We used the validation set of CNNDM as the validation set of GIGA-CM as well. As in See et al. (2017), documents and summaries in CNNDM, NYT50 and GIGA-CM are all segmented and tokenized using Stanford CoreNLP toolkit (Manning et al., 2014). To reduce the vocabulary size, we applied byte pair encoding (BPE; Sennrich et al. 2016) to all of our datasets. To limit the memory consumption during training, we limit the length of each sentence to be 50 words (51th word and onwards are removed) and split documents with more than 30 sentences into smaller documents with each containing at most 30 sentences.

## 4.2 Implementation Details

Our model is trained in three stages, which includes two pre-training stages and one finetuning stage. The first stage is the open-domain pre-training and in this stage we train HIBERT with the pre-training objective (Section 3.2) on GIGA-CM dataset. In the second stage, we perform the in-domain pre-training on the CNNDM (or NYT50) dataset still with the same pre-training objective. In the final stage, we finetune HIBERT in the summarization model (Section 3.3) to predict extractive sentence labels on CNNDM (or NYT50).

The sizes of the sentence and document level Transformers as well as the Transformer decoder in HIBERT are the same. Let $L$ denote the number of layers in Transformer, $H$ the hidden size and $A$ the number of attention heads. As in (Vaswani et al., 2017; Devlin et al., 2018), the hidden size of the feedforward sublayer is $4H$. We mainly trained two model sizes: HIBERT$_S$ ($L = 6$, $H = 512$ and $A = 8$) and HIBERT$_M$ ($L = 6$, $H = 768$ and $A = 12$). We trained both HIBERT$_S$ and HIBERT$_M$ on a single machine with 8 Nvidia Tesla V100 GPUs with a batch size of 256 documents. We optimized our models using Adam with learning rate of 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 norm of 0.01, learning rate warmup 10,000 steps and learning rate decay afterwards using the strategies in Vaswani et al. (2017). The dropout rate in all layers are 0.1. In pre-training stages, we trained our models until validation perplexities do not decrease significantly (around 45 epochs on GIGA-CM dataset and 100 to 200 epochs on CNNDM and NYT50). Training HIBERT$_M$ for one epoch on GIGA-CM dataset takes approximately 20 hours.

Our models during fine-tuning stage can be trained on a single GPU. The hyper-parameters are almost identical to these in the pre-training stages except that the learning rate is 5e-5, the batch size is 32, the warmup steps are 4,000 and we train our models for 5 epochs. During inference, we rank sentences using $p(y_i|\mathcal{D})$ (Equation (8)) and choose the top $K$ sentences as summary, where $K$ is tuned on the validation set.

## 4.3 Evaluations

We evaluated the quality of summaries from different systems automatically using ROUGE (Lin, 2004). We reported the full length F1 based ROUGE-1, ROUGE-2 and ROUGE-L on the

---

[3]Scripts publicly available at https://github.com/abisee/cnn-dailymail

[4]https://catalog.ldc.upenn.edu/LDC2012T21

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| Pointer+Coverage | 39.53 | 17.28 | 36.38 |
| Abstract-ML+RL | 39.87 | 15.82 | 36.90 |
| DCA | 41.69 | 19.47 | 37.92 |
| SentRewrite | 40.88 | 17.80 | 38.54 |
| InconsisLoss | 40.68 | 17.97 | 37.13 |
| Bottom-Up | 41.22 | 18.68 | 38.34 |
| Lead3 | 40.34 | 17.70 | 36.57 |
| SummaRuNNer | 39.60 | 16.20 | 35.30 |
| NeuSum | 40.11 | 17.52 | 36.39 |
| Refresh | 40.00 | 18.20 | 36.60 |
| NeuSum-MMR | 41.59 | 19.01 | 37.98 |
| BanditSum | 41.50 | 18.70 | 37.60 |
| JECS | 41.70 | 18.50 | 37.90 |
| LatentSum | 41.05 | 18.77 | 37.54 |
| HierTransformer | 41.11 | 18.69 | 37.53 |
| BERT | 41.82 | 19.48 | 38.30 |
| $\text{HIBERT}_S$ (in-domain) | 42.10 | 19.70 | 38.53 |
| $\text{HIBERT}_S$ | **42.31** | **19.87** | **38.78** |
| $\text{HIBERT}_M$ | **42.37** | **19.95** | **38.83** |

Table 1: Results of various models on the CNNDM test set using full-length F1 ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L).

CNNDM and NYT50 datasets. We compute ROUGE scores using the `ROUGE-1.5.5.pl` script.

Additionally, we also evaluated the generated summaries by eliciting human judgments. Following (Cheng and Lapata, 2016; Narayan et al., 2018), we randomly sampled 20 documents from the CNNDM test set. Participants were presented with a document and a list of summaries produced by different systems. We asked subjects to rank these summaries (ties allowed) by taking informativeness (is the summary capture the important information from the document?) and fluency (is the summary grammatical?) into account. Each document is annotated by three different subjects.

## 4.4 Results

Our main results on the CNNDM dataset are shown in Table 1, with abstractive models in the top block and extractive models in the bottom block. Pointer+Coverage (See et al., 2017), Abstract-ML+RL (Paulus et al., 2017) and DCA (Celikyilmaz et al., 2018) are all sequence to sequence learning based models with copy and coverage modeling, reinforcement learning and deep communicating agents extensions. SentRewrite

(Hsu et al., 2018) and InconsisLoss (Chen and Bansal, 2018) all try to decompose the word by word summary generation into sentence selection from document and "sentence" level summarization (or compression). Bottom-Up (Gehrmann et al., 2018) generates summaries by combines a word prediction model with the decoder attention model. The extractive models are usually based on hierarchical encoders (SummaRuNNer; Nallapati et al. 2017 and NeuSum; Cheng and Lapata 2016). They have been extended with reinforcement learning (Refresh; Narayan et al. 2018 and BanditSum; Dong et al. 2018), Maximal Marginal Relevance (NeuSum-MMR; Zhou et al. 2018), latent variable modeling (LatentSum; Zhang et al. 2018) and syntactic compression (JECS; Xu and Durrett 2019). Lead3 is a baseline which simply selects the first three sentences. Our model $\text{HIBERT}_S$ (in-domain), which only use one pre-training stage on the in-domain CNNDM training set, outperforms all of them and differences between them are all significant with a 0.95 confidence interval (estimated with the ROUGE script). Note that pre-training $\text{HIBERT}_S$ (in-domain) is very fast and it only takes around 30 minutes for one epoch on the CNNDM training set. Our models with two pre-training stages ($\text{HIBERT}_S$) or larger size ($\text{HIBERT}_M$) perform even better and $\text{HIBERT}_M$ outperforms BERT by 0.5 ROUGE[5]. We also implemented two baselines. One is the hierarchical transformer summarization model (HeriTransfomer; described in 3.3) without pre-training. Note the setting for HeriTransfomer is ($L = 4, H = 300$ and $A = 4$) [6]. We can see that the pre-training (details in Section 3.2) leads to a +1.25 ROUGE improvement. Another baseline is based on a pre-trained BERT (Devlin et al., 2018)[7] and finetuned on the CNNDM dataset. We used the $\text{BERT}_{\text{base}}$ model because our 16G RAM V100 GPU cannot fit $\text{BERT}_{\text{large}}$ for the summarization task even with batch size of 1. The positional embedding of BERT supports input length up to 512 words, we therefore split documents with more than 10 sentences into multiple blocks

---

[5]The difference is significant according to the ROUGE script.

[6]We tried deeper and larger models, but obtained inferior results, which may indicates training large or deep models on this dataset without a good initialization is challenging.

[7]Our BERT baseline is adapted from this implementation https://github.com/huggingface/pytorch-pretrained-BERT

| Models | R-1 | R-2 | R-L |
|---|---|---|---|
| Lead | 41.80 | 22.60 | 35.00 |
| EXTRACTION | 44.30 | 25.50 | 37.10 |
| JECS | 45.50 | 25.30 | 38.20 |
| HeriTransformer | 47.44 | 28.08 | 39.56 |
| BERT | 48.38 | 29.04 | 40.53 |
| HIBERT$_S$ (in-domain) | 48.92 | 29.58 | 41.10 |
| HIBERT$_M$ (in-domain) | 49.06 | 29.70 | 41.23 |
| HIBERT$_S$ | **49.25** | **29.92** | **41.43** |
| HIBERT$_M$ | **49.47** | **30.11** | **41.63** |

Table 2: Results of various models on the NYT50 test set using full-length F1 ROUGE. HIBERT$_S$ (in-domain) and HIBERT$_M$ (in-domain) only uses one pre-training stage on the NYT50 training set.

| Pretraining Strategies | R-1 | R-2 | R-L |
|---|---|---|---|
| Open-Domain | 42.97 | 20.31 | 39.51 |
| In-Domain | 42.93 | 20.28 | 39.46 |
| Open+In-Domain | **43.19** | **20.46** | **39.72** |

Table 3: Results of summarization model (HIBERT$_S$ setting) with different pre-training strategies on the CNNDM validation set using full-length F1 ROUGE.

(each block with 10 sentences[8]). We feed each block (the `BOS` and `EOS` tokens of each sentence are replaced with `[CLS]` and `[SEP]` tokens) into BERT and use the representation at `[CLS]` token to classify each sentence. Our model HIBERT$_S$ outperforms BERT by 0.4 to 0.5 ROUGE despite with only half the number of model parameters (HIBERT$_S$ 54.6M v.s. BERT 110M).

Results on the NYT50 dataset show the similar trends (see Table 2). EXTRACTION is a extractive model based hierarchical LSTM and we use the numbers reported by Xu and Durrett (2019). The improvement of HIBERT$_M$ over the baseline without pre-training (HeriTransformer) becomes 2.0 ROUGE. HIBERT$_S$ (in-domain), HIBERT$_M$ (in-domain), HIBERT$_S$ and HIBERT$_M$ all outperform BERT significantly according to the ROUGE script.

We also conducted human experiment with 20 randomly sampled documents from the CNNDM test set. We compared our model HIBERT$_M$ against Lead3, DCA, Latent, BERT and the human reference (Human)[9]. We asked the subjects to rank

---

[8]We use 10 sentences per block, because maximum sentence length $50 \times 10 < 512$ (maximum BERT supported length). The last block of a document may have less than 10 sentences.

[9]We obtained the outputs of DCA via emails.

| Models | 1st | 2nd | 3rd | 4th | 5th | 6th | MeanR |
|---|---|---|---|---|---|---|---|
| Lead3 | 0.03 | 0.18 | 0.15 | 0.30 | 0.30 | 0.03 | 3.75 |
| DCA | 0.08 | 0.15 | 0.18 | 0.20 | 0.15 | 0.23 | 3.88 |
| Latent | 0.05 | 0.33 | 0.28 | 0.20 | 0.13 | 0.00 | 3.03 |
| BERT | 0.13 | 0.37 | 0.32 | 0.15 | 0.03 | 0.00 | 2.58 |
| HIBERT$_M$ | 0.30 | 0.35 | 0.25 | 0.10 | 0.00 | 0.00 | 2.15 |
| Human | 0.58 | 0.15 | 0.20 | 0.00 | 0.03 | 0.03 | 1.85 |

Table 4: Human evaluation: proportions of rankings and mean ranks (MeanR; lower is better) of various models.

the outputs of these systems from best to worst. As shown in Table 4, the output of HIBERT$_M$ is selected as the best in 30% of cases and we obtained lower mean rank than all systems except for Human. We also converted the rank numbers into ratings (rank $i$ to $7 - i$) and applied student $t$-test on the ratings. HIBERT$_M$ is significantly different from all systems in comparison ($p < 0.05$), which indicates our model still lags behind Human, but is better than all other systems.

**Pre-training Strategies** As mentioned earlier, our pre-training includes two stages. The first stage is the open-domain pre-training stage on the GIGA-CM dataset and the following stage is the in-domain pre-training on the CNNDM (or NYT50) dataset. As shown in Table 3, we pretrained HIBERT$_S$ using only open-domain stage (Open-Domain), only in-domain stage (In-Domain) or both stages (Open+In-Domain) and applied it to the CNNDM summarization task. Results on the validation set of CNNDM indicate the two-stage pre-training process is necessary.

## 5 Conclusions

The core part of a neural extractive summarization model is the hierarchical document encoder. We proposed a method to pre-train document level hierarchical bidirectional transformer encoders on unlabeled data. When we only pre-train hierarchical transformers on the training sets of summarization datasets with our proposed objective, application of the pre-trained hierarchical transformers to extractive summarization models already leads to wide improvement of summarization performance. Adding the large open-domain dataset to pre-training leads to even better performance. In the future, we plan to apply models to other tasks that also require hierarchical document encodings (e.g., document question answering). We are also interested in improving the architectures

of hierarchical document encoders and designing other objectives to train hierarchical transformers.

## Acknowledgments

## References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *In Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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)*, pages 1662–1675, New Orleans, Louisiana.

Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4088–4097. Association for Computational Linguistics.

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)*, pages 675–686. Association for Computational Linguistics.

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)*, pages 484–494, Berlin, Germany.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

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. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008. Association for Computational Linguistics.

Elena Filatova and Vasileios Hatzivassiloglou. 2004a. Event-based extractive summarization. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 104–111, Barcelona, Spain.

Elena Filatova and Vasileios Hatzivassiloglou. 2004b. Event-based extractive summarization.

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. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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. Curran Associates, Inc.

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

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Pub Co.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *In Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3075–3091, San Francisco, California.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

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)*, pages 1747–1759, New Orleans, Louisiana.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580. ACM.

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. 2018. 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)*, pages 2227–2237. Association for Computational Linguistics.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. Mead - a platform for multidocument multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

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)*, pages 1073–1083, Vancouver, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

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.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. *arXiv preprint arXiv:1902.00863*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784. Association for Computational Linguistics.

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)*, pages 654–663. Association for Computational Linguistics.