Unsupervised Extractive Summarization by Pre-training Hierarchical Transformers

Shusheng Xu[1]*, **Xingxing Zhang**[2], **Yi Wu**[1,3], **Furu Wei**[2] and **Ming Zhou**[2]

[1] IIIS, Tsinghua Unveristy, Beijing, China
[2] Microsoft Research Asia, Beijing, China
[3] Shanghai Qi Zhi institute, Shanghai China

xuss20@mails.tsinghua.edu.cn
{xizhang,fuwei,mingzhou}@microsoft.com
jxwuyi@gmail.com

Abstract

Unsupervised extractive document summarization aims to select important sentences from a document without using labeled summaries during training. Existing methods are mostly graph-based with sentences as nodes and edge weights measured by sentence similarities. In this work, we find that transformer attentions can be used to rank sentences for unsupervised extractive summarization. Specifically, we first pre-train a hierarchical transformer model using unlabeled documents only. Then we propose a method to rank sentences using sentence-level self-attentions and pre-training objectives. Experiments on CNN/DailyMail and New York Times datasets show our model achieves state-of-the-art performance on unsupervised summarization. We also find in experiments that our model is less dependent on sentence positions. When using a linear combination of our model and a recent unsupervised model explicitly modeling sentence positions, we obtain even better results.

1 Introduction

Document summarization is the task of transforming a long document into its shorter version while still retaining its important content. Researchers have explored many paradigms for summarization, while the most popular ones are *extractive* summarization and *abstractive* summarization (Nenkova and McKeown, 2011). As their names suggest, extractive summarization generates summiries by extracting text from original documents, and abstractive summarization rewrites documents by paraphrasing or deleting some words or phrases.

Most summarization models require labeled data, where documents are paired with human written summaries. Unfortunately, human labeling for summarization task is expensive and therefore high quality large scale labeled summarization datasets are rear (Hermann et al., 2015) compared to growing web documents created everyday. It is also not possible to create summaries for documents in all text domains and styles. In this paper, we focus on unsupervised summarization, where we only need unlabeled documents during training.

Many attempts for unsupervised summarization are extractive (Carbonell and Goldstein, 1998; Radev et al., 2000; Lin and Hovy, 2002; Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Wan, 2008; Wan and Yang, 2008; Hirao et al., 2013; Parveen et al., 2015). The core problem is to identify salient sentences in a document. The most popular approaches among these work rank sentences in the document using graph based algorithms, where each node is a sentence and weights of edges are measured by sentence similarities. Then a graph ranking method is employed to estimate sentence importance. For example, TextRank (Mihalcea and Tarau, 2004) utilizes word co-occurrence statistics to compute similarity and then employs PageRank (Page et al., 1997) to rank sentences. Sentence similarities in (Zheng and Lapata, 2019) are measured with BERT (Devlin et al., 2019) and sentences are sorted w.r.t. their centralities in a directed graph.

Recently, there has been increasing interest in developing unsupervised abstractive summarization models (Wang and Lee, 2018; Fevry and Phang, 2018; Chu and Liu, 2019; Yang et al., 2020). These models are mostly based on sequence to sequence learning (Sutskever et al., 2014) and sequential denoising auto-encoding (Dai and Le, 2015). Unfortunately, there is no guarantee that summaries produced by these models are grammatical and consistent with facts described original documents.

Zhang et al. (2019) propose an unsupervised method to pre-train a hierarchical transformer model (i.e., HIBERT) for document modeling. The

* Work done during the first author's internship at Microsoft Research Asia.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1784–1795
November 16 - 20, 2020.

hierarchical transformer has a token-level transformer to learn sentence representations and a sentence-level transformer to learn interactions between sentences with self-attention. In Zhang et al. (2019), HIBERT is applied to supervised extractive summarization. However, we believe that after pre-training HIBERT on large scale unlabeled data, the self-attention scores in the sentence-level transformer becomes meaningful for estimating the importance of sentences. Intuitively, if many sentences in a document attend to one particular sentence with high attention scores, then this sentence should be important. In this paper, we find that (sentence-level) transformer attentions (in a hierarchical transformer) can be used to rank sentences for unsupervised extractive summarization, while previous work mostly leverage graph based (or rule based) methods and sentence similarities computed with off-the-shelf sentence embeddings. Specifically, we first introduce two pre-training tasks for hierarchical transformers (i.e., extended HIBERT) to obtain sentence-level self-attentions using unlabled documents only. Then, we design a method to rank sentences by using sentence-level self-attentions and pre-training objectives. Experiments on CNN/DailyMail and New York Times datasets show our model achieves state-of-the-art performance on unsupervised summarization. We also find in experiments that our model is less dependent on sentence positions. When using a linear combination of our model and a recent unsupervised model explicitly modeling sentence positions, we obtain even better results. Our code and models are available at https://github.com/xssstory/STAS.

2 Related Work

In this section, we introduce work on supervised summarization, unsupervised summarization and pre-training.

Supervised Summarization Most summarization models require supervision from labeled datasets, where documents are paired with human written summaries. As mentioned earlier, extractive summarization aims to extract important sentences from documents and it is usually viewed as a (sentence) ranking problem by using scores from classifiers (Kupiec et al., 1995) or sequential labeling models (Conroy and O'leary, 2001). Summarization performance of this class of methods are greatly improved, when human engineered features (Radev et al., 2004; Nenkova et al., 2006; Filatova and Hatzivassiloglou, 2004) are replaced with convolutional neural networks (CNN) and long short-term memory networks (LSTM) (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018; Zhang et al., 2018).

Abstractive summarization on the other hand can generate new words or phrases and are mostly based on sequence to sequence (seq2seq) learning (Bahdanau et al., 2015). To better fit in the summarization task, the original seq2seq model is extended with copy mechanism (Gu et al., 2016), coverage model (See et al., 2017), reinforcement learning (Paulus et al., 2018) as well as bottom-up attention (Gehrmann et al., 2018). Recently, pre-trained transformers (Vaswani et al., 2017) achieve tremendous success in many NLP tasks (Devlin et al., 2019; Liu et al., 2019). Pre-trained methods customized for both extractive (Zhang et al., 2019; Liu and Lapata, 2019) and abstractive (Dong et al., 2019; Lewis et al., 2019) summarization again advance the state-of-the-art in supervised summarization. Our model also leverages pre-trained methods and models, but it is unsupervised.

Unsupervised Summarization Compared to supervised models, unsupervised models only need unlabeled documents during training. Most unsupervised extractive models are graph based (Carbonell and Goldstein, 1998; Radev et al., 2000; Lin and Hovy, 2002; Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Wan, 2008; Wan and Yang, 2008; Hirao et al., 2013; Parveen et al., 2015). For example, TextRank (Mihalcea and Tarau, 2004) treats sentences in a document as nodes in an undirected graph, and edge weights are measured with co-occurrence based similarities between sentences. Then PageRank (Page et al., 1999) is employed to determine the final ranking scores for sentences. Zheng and Lapata (2019) builds directed graph by utilizing BERT (Devlin et al., 2019) to compute sentence similarities. The importance score of a sentence is the weighted sum of all its out edges, where weights for edges between the current sentence and preceding sentences are negative. Thus, leading sentences tend to obtain high scores. Unlike Zheng and Lapata (2019), sentence positions are not explicitly modeled in our model and therefore our model is less dependent on sentence positions (as shown in experiments).

There are also an interesting line of work on unsupervised abstractive summarization. Yang et al.

(2020) pre-trains a seq2seq Transformer by predicting the first three sentences of news documents and then further tunes the model with semantic classification and denoising auto-encoding objectives. The model described in Wang and Lee (2018) utilizes seq2seq auto-encoding coupled with adversarial training and reinforcement learning. Fevry and Phang (2018) and Baziotis et al. (2019) focus on sentence summarization (i.e., compression). Chu and Liu (2019) proposes yet another denoising auto-encoding based model in multi-document summarization domain. However, the performance of these unsupervised models are still unsatisfactory compared to their extractive counterparts.

Pre-training Pre-training methods in NLP learn to encode text by leveraging unlabeled text. Early work mostly concentrate on pre-training word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017). Later, sentence encoder can also be pre-trained with language model (or masked language model) objectives (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019). Zhang et al. (2019) propose a method to pre-train a hierarchical transformer encoder (document encoder) by predicting masked sentences in a document for *supervised summarization*, while we focus on *unsupervised summarization*. In our method, we also propose a new task (sentence shuffling) for pre-training hierarchical transformer encoders. Iter et al. (2020) propose a contrastive pre-training objective to predict relative distances of surrounding sentences to the anchor sentence, while our sentence shuffling task predicts original positions of sentences from a shuffled docuemt. Besides, pre-training methods mentioned above focus on learning good word, sentence or document representations for downstream tasks, while our method focuses on learning sentence level attention distributions (i.e., sentence associations), which is shown in our experiments to be very helpful for unsupervised summarization.

3 Model

In this section, we describe our unsupervised summarization model STAS (as shorthand for **S**entence-level **T**ransformer based **A**ttentive **S**ummarization). We first introduce how documents are encoded in our model. Then we present methods to pre-trained our document encoder. Finally we apply the pre-trained encoder to unsupervised summarization.

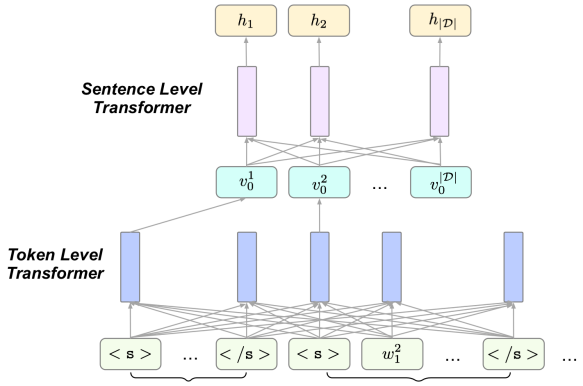

Figure 1: The architecture of our hierarchical encoder, the token level Transformer encodes tokens and then the sentence level Transformer learns final sentence representations from representations at <s>.

3.1 Document Modeling

Let $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ denote a document, where $S_i = (w_0^i, w_1^i, w_2^i, \ldots, w_{|S_i|}^i)$ is a sentence in $\mathcal{D}$ and w_j^i is a token in S_i. As a common wisdom, we also add two special tokens (i.e., w_0^i = <s> and $w_{|S_i|}^i$ = </s>) to S_i, which represents the begin and end of a sentence, respectively. Transformer models (Vaswani et al., 2017), which are composed of multiple self-attentive layers and skip connections (He et al., 2016), have shown tremendous success in text encoding (Devlin et al., 2019). Due to the hierarchical nature of documents, we encode the document $\mathcal{D}$ using a hierarchical Transformer encoder, which contains a *token-level* Transformer $Trans^T$ and a *sentence-level* Transformer $Trans^S$ as shown in Figure 1. Let $||$ denote an operator for sequences concatenation. $Trans^T$ views $\mathcal{D}$ as a *flat* sequence of tokens denoted as $D = (S_1 \,||\, S_2 \,||\, \ldots \,||\, S_{|\mathcal{D}|})$. After we apply $Trans^T$ to D, we obtain contextual representations for all tokens $(\mathbf{v}_0^1, \mathbf{v}_1^1, \ldots, \mathbf{v}_{|S_1|}^1, \ldots, \mathbf{v}_j^i, \ldots, \mathbf{v}_0^{|\mathcal{D}|}, \ldots, \mathbf{v}_{|S_{|\mathcal{D}|}|}^{|\mathcal{D}|})$. We use the representation at each <s> token as the representation for that sentence and therefore representations for all sentences in $\mathcal{D}$ are $\mathbf{V} = (\mathbf{v}_0^1, \mathbf{v}_0^2, \ldots, \mathbf{v}_0^{|\mathcal{D}|})$. The *sentence-level* Transformer $Trans^S$ takes $\mathbf{V}$ as input and learns sentence representations given other sentences in $\mathcal{D}$ as context:

$$\mathbf{H}, \mathbf{A} = Trans^S(\mathbf{V}) \tag{1}$$

where $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{|\mathcal{D}|})$ and $\mathbf{h}_i$ is the final representation of S_i; $\mathbf{A}$ is the self-attention matrix

and $\mathbf{A}_{i,j}$ is the attention score from sentence S_i to sentence S_j. $Trans^S$ contains multiple layers and each layer contains multiple attention heads. To obtain $\mathbf{A}$, we first average the attention scores across different heads and then across different layers. Our hierarchical document encoder is similar to the hierarchical Transformer model described in Zhang et al. (2019). The main difference is that our *token-level* Transformer encodes all sentences in a document as a whole rather than separately.

3.2 Pre-training

In this section, we pre-train the hierarchical document encoder introduced in Section 3.1 using unlabeled documents only. We expect that after pre-training, the encoder would obtain the ability of modeling interactions (i.e., attentions) among sentences in a document. In this following, we introduce two tasks we used to pre-train the encoder.

Masked Sentences Prediction The first task is Masked Sentences Prediction (MSP) described in Zhang et al. (2019). We randomly mask 15% of sentences in a document and then predict the original sentences. Let $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ denote a document and $\widetilde{\mathcal{D}} = (\widetilde{S}_1, \ldots, \widetilde{S}_{|\mathcal{D}|})$ the document with some sentences masked, where

$$\widetilde{S}_i = \begin{cases} S_i & 85\% \quad \text{of cases} \\ mask(S_i) & 15\% \quad \text{of cases} \end{cases} \tag{2}$$

$mask(S_i)$ is a function to mask S_i, which in 80% of cases replaces each word in S_i with the `[MASK]` token, in 10% of cases replaces S_i with a random sentence and in the remaining 10% of cases keep S_i unchanged. The masking strategy is similar to that of BERT (Devlin et al., 2019), but it is applied on sentence level. Let $\mathcal{I} = \{i | \widetilde{S}_i = mask(S_i)\}$ denote the set of indices for masked sentences and $\mathcal{O} = \{S_i | i \in \mathcal{I}\}$ the original sentences corresponding to masked sentences.

Supposing $i \in \mathcal{I}$, we demonstrate how we predict the original sentence $S_i = (w_0^i, w_1^i, \ldots, w_{|S_i|}^i)$ given $\widetilde{\mathcal{D}}$. As shown in Figure 2, we first encode $\widetilde{\mathcal{D}}$ using the encoder in Section 3.1 and obtain $\widetilde{\mathbf{H}} = (\widetilde{\mathbf{h}}_1, \widetilde{\mathbf{h}}_2, \ldots, \widetilde{\mathbf{h}}_{|\mathcal{D}|})$. Then we use $\widetilde{\mathbf{h}}_i$ (i.e., the contextual representation of $\widetilde{S}_i$) to predict S_i one token at a time with a conditional Transformer decoder $TransDec^M$. We inject the information of $\widetilde{S}_i$ to $TransDec^M$ by adding $\widetilde{\mathbf{h}}_i$ after the self attention sub-layer of each Transformer block in

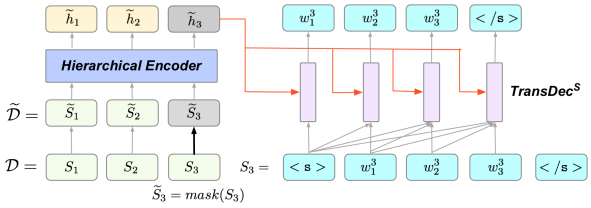

Figure 2: An example of masked sentences prediction. The third sentence in the document is masked and the hierarchical encoder encodes the masked document. We then use $TransDec^S$ to predict the original sentence one token at a time.

$TransDec^M$. Assuming $w_{0:j-1}^i$ has been generated, the probability of w_j^i given $w_{0:j-1}^i$ and $\widetilde{\mathcal{D}}$ is

$$\hat{\mathbf{h}}_j^i = TransDec^M(w_{0:j-1}^i, \widetilde{\mathbf{h}}_i) \tag{3}$$

$$p(w_j^i | w_{0:j-1}^i, \widetilde{\mathcal{D}}) = \text{softmax}(\mathbf{W}_o \hat{\mathbf{h}}_j^i) \tag{4}$$

The probability of all original sentences given $\widetilde{\mathcal{D}}$ is

$$p(\mathcal{O} | \widetilde{\mathcal{D}}) = \prod_{S_i \in \mathcal{O}} \prod_{j=1}^{|S_i|} p(w_j^i | w_{0:j-1}^i, \widetilde{\mathcal{D}}) \tag{5}$$

MSP is proposed in HIBERT (Zhang et al., 2019) for *supervised summarization*, while we use MSP and transformer attention for sentence ranking in *unsupervised summarization* (Section 3.3). Note that the goal and the way of using MSP in this work is different from these in HIBERT.

Sentence Shuffling We propose a new task that shuffles the sentences in a document and then select sentences in the original order one by one. We expect that the hierarchical document encoder can learn to select sentences based on their contents rather than positions.

Recall that $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ is a document, we shuffle the sentences in $\mathcal{D}$ and obtain a permuted document $\mathcal{D}' = (S'_1, S'_2, \ldots, S'_{|\mathcal{D}|})$ where S_i is the ith sentence in the original document and there exists a sentence $S'_{P_i} = S_i$ in the permuted document $\mathcal{D}'$ (i.e., $P_i \in [1, |\mathcal{D}|]$ is the position of S_i in $\mathcal{D}'$). In this task, we predict $\mathcal{P} = (P_1, P_2, \ldots, P_{|\mathcal{D}|})$.

As shown in Figure 3, we first use the document encoder in Section 3.1 to encode $\mathcal{D}'$ and yields its context dependent sentence representations $\mathbf{H}' = (\mathbf{h}'_1, \mathbf{h}'_2, \ldots, \mathbf{h}'_{|\mathcal{D}|})$. Supposing that $P_0, P_1, P_2, \ldots, P_{t-1}$ are known[1], we predict P_t using a Pointer Network (Vinyals et al., 2015) with

[1]We set $P_0 = 0$; $\mathbf{h}'_0$ is a zero vector.

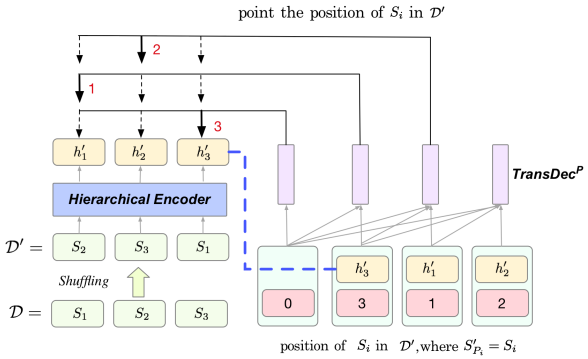

Figure 3: An example of Sentence Shuffling. The sentences in the document are shuffled and then pass through the hierarchical encoder, then a Pointer Network with $TransDec^P$ as its decoder is adopted to predict the positions of original sentences in the shuffled document.

Transformer as its decoder. Let $TransDec^P$ denote the transformer decoder in PointerNet, $\mathbf{E}_{P_i}$ is the *absolute* positional embedding of P_i in original document and $\mathbf{p}_i$ the positional embedding of P_i during decoding. The input of $TransDec^P$ is the sum of sentence representations and positional embeddings:

$$\mathbf{M}_{t-1} = (\mathbf{h}'_{P_1} + \mathbf{p}_1 + \mathbf{E}_{P_1}, \ldots, \mathbf{h}'_{P_{t-1}} + \mathbf{p}_{t-1} + \mathbf{E}_{P_{t-1}})$$

The output $\mathbf{h}^o_t$ summaries the sentences *depermutated* so far.

$$\mathbf{h}^o_t = TransDec^P(\mathbf{M}_{t-1}) \quad (6)$$

Then the probability of selecting S'_{p_t} is estimated with the attention (Bahdanau et al., 2015) between $\mathbf{h}^o_t$ and all sentences in $\mathcal{D}'$ as follows:

$$p(P_t|P_{1:t-1}, \mathcal{D}') = \frac{\exp(g(\mathbf{h}^o_t, \mathbf{h}'_{P_t}))}{\sum_i \exp(g(\mathbf{h}^o_t, \mathbf{h}'_i))} \quad (7)$$

where g is a feed forward neural network with the following parametrization:

$$g(\mathbf{h}^o_t, \mathbf{h}'_i) = \mathbf{v}_a^\top \tanh(\mathbf{U}_a \mathbf{h}^o_t + \mathbf{W}_a \mathbf{h}'_i) \quad (8)$$

where $\mathbf{v}_a \in \mathbb{R}^{d\times 1}$, $\mathbf{U}_a \in \mathbb{R}^{d\times d}$, $\mathbf{W}_a \in \mathbb{R}^{d\times d}$ are trainable parameters. Finally the probability of positions of original sentences in the shuffled document is:

$$p(\mathcal{P}|\mathcal{D}') = \prod_{t=1}^{|\mathcal{D}|} p(P_t|P_{0:t-1}, \mathcal{D}') \quad (9)$$

During training, for each batch of documents we apply both the masked sentence prediction and sentence shuffling tasks. One document $\mathcal{D}$ generates a masked document $\widetilde{\mathcal{D}}$ and a shuffled document $\mathcal{D}'$. Note that 15% of sentences are masked in the masked document $\widetilde{\mathcal{D}}$, and all sentences are shuffled in the shuffled document $\mathcal{D}'$. The whole model is optimized with the following objective:

$$L(\theta) = -\sum_{\mathcal{D}\in\mathcal{X}} \log p(\mathcal{O}|\widetilde{\mathcal{D}}) + \log p(\mathcal{P}|\mathcal{D}')$$

where $\mathcal{D}$ is a document in the training document set $\mathcal{X}$.

3.3 Unsupervised Summarization

In this section, we propose our unsupervised extractive summarization method. Extractive summarization aims to select the most important sentences in document. Once we have obtained a hierarchical encoder using the pre-training methods in Section 3.2, we are ready to rank sentences and no additional fine-tuning is needed in this step.

Our first ranking criteria is based on the probabilities of sentences in a document. Recall that $\mathcal{D} = (S_1, S_2, \ldots, S_{|\mathcal{D}|})$ is a document and its probability is

$$p(\mathcal{D}) = \prod_{i=1}^{|\mathcal{D}|} p(S_i|S_{1:i-1}) \approx \prod_{i=1}^{|\mathcal{D}|} p(S_i|\mathcal{D}_{\neg S_i}) \quad (10)$$

It is not straight forward to estimate $p(S_i|S_{1:i-1})$ directly since document models in this work are all bidirectional. However, we can estimate $p(S_i|\mathcal{D}_{\neg S_i})$ using the masked sentences prediction task in Section 3.2. We therefore use $p(S_i|\mathcal{D}_{\neg S_i})$ to approximate $p(S_i|S_{1:i-1})$. Finding the most important sentence is equivalent to finding the sentence with highest probability (i.e., $p(S_i|\mathcal{D}_{\neg S_i})$). In the following we demonstrate how to estimate $p(S_i|\mathcal{D}_{\neg S_i})$. As in Section 3.2, we create $\mathcal{D}_{\neg S_i}$ by masking S_i in $\mathcal{D}$ (i.e., replacing all tokens in S_i with `[MASK]` tokens). $p(S_i|\mathcal{D}_{\neg S_i})$ can be estimated using Equation (5). To make the probabilities of different sentences comparable, we normalize them by their length. Then we obtain $\hat{r}_i$ as follows[2] (also see Equation (5))

$$\hat{r}_i = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} p(w^i_j|w^i_{0:j-1}, \mathcal{D}_{\neg S_i}) \quad (11)$$

We also normalize $\hat{r}_i$ across sentences (in a document) and obtain our first ranking criteria $\widetilde{r}_i$:

$$\widetilde{r}_i = \frac{\hat{r}_i}{\sum_{j=1}^{|\mathcal{D}|} \hat{r}_j} \quad (12)$$

[2]We also tried the geometric average, but the effect is not as good as the arithmetic average.

In the second ranking criteria, we model the contributions of other sentences to the current sentence explicitly. We view a document $\mathcal{D}$ as a directed graph, where each sentence in it is a node. The connections between sentences (i.e., edge weights) can be modeled using the self-attention matrix $\mathbf{A}$ of the sentence level Transformer encoder described in Section 3.1, which is produced by a pre-trained hierarchical document encoder. We assume that a sentence S_j can transmit its importance score $\widetilde{r}_i$ to an arbitrary sentence S_i through the edge between them. Let $\mathbf{A}_{j,i}$ denote the attention score from S_j to S_i. After receiving all transmissions from all sentences, the second ranking score for S_i is as follows:

$$r'_i = \sum_{j=1, j\neq i}^{|\mathcal{D}|} \mathbf{A}_{j,i} \times \widetilde{r}_j \tag{13}$$

The final ranking score of S_i combines the score from itself as well as other sentences:

$$r_i = \gamma_1 \, \widetilde{r}_i + \gamma_2 \, r'_i \tag{14}$$

γ_1 and γ_2 are coefficients tuned on development set. r_i can be computed iteratively by assigning r_i to $\widetilde{r}_i$ and repeating Equation (13) and Equation (14) for T iterations. We find a small T ($T \leq 3$) works well according to the development set.

4 Experiments

In this section we assess the performance of STAS on the document summarization task. We firstly introduce datasets we used and then give our implementation details. Finally we compare our method against previous methods.

4.1 Datasets

We evaluate STAS on two summarization datasets, namely the CNN/DailyMail (CNN/DM; Hermann et al. 2015) dataset and the New York Times (NYT; Sandhaus 2008) dataset. CNN/DM is composed of articles from CNN and Daily Mail news websites, which uses their associated highlights as reference summaries. NYT dataset contains articles published by the New York Times between January 1, 1987 and June 19, 2007 and summaries are written by library scientists. For the CNN/DM dataset, we follow the standard splits and preprocessing steps used in supervised summarization[3] (See et al., 2017; Liu and Lapata, 2019), and the resulting dataset contains 287,226 articles for training, 13,368 for validation and 11,490 for test. Following Zheng and Lapata (2019), we adopted the splits widely used in abstractive summarization (Paulus et al., 2018) for the NYT dataset, which ranks articles by their publication date and used the first 589,284 for training, the next 32,736 for validation and the remaining 32,739 for test. Then, we filter out documents whose summaries are shorter than 50 words as in (Zheng and Lapata, 2019) and finally retain 36,745 for training, 5,531 for validation and 4,375 for test.

We segment sentences using the Stanford CoreNLP toolkit (Manning et al., 2014). Sentences are then tokenized with the UTF-8 based BPE tokenizer used in RoBERTa and GPT-2 (Radford et al., 2019) and the resulting vocabulary contains 50,265 subwords. During training, we only leverage articles in CNN/DM or NYT; while we do use both articles and summaries in validation sets to tune hyper-parameters of our models.

We evaluated the quality of summaries from different models using ROUGE (Lin, 2004). We report the full length F1 based ROUGE-1, ROUGE-2, ROUGE-L on both CNN/DM and NYT datasets. These ROUGE scores are computed using the `ROUGE-1.5.5.pl` script[4].

4.2 Implementation Details

The main building blocks of STAS are Transformers (Vaswani et al., 2017). In the following, we describe the sizes of them using the number of layers L, the number of attention heads A, and the hidden size N. As in (Vaswani et al., 2017; Devlin et al., 2019), the hidden size of the feed-forward sublayer is always $4H$. STAS contains one hierarchical encoder (see Section 3.1) and two decoders, where they are used for the masked sentences prediction and sentence shuffling pre-training tasks (see Section 3.2). The token-level encoder is initialized with the parameters of RoBERTa$_{\text{BASE}}$ (Liu et al., 2019)[5] and we set $L = 12, H = 768, A = 12$. The sentence-level encoder and the two decoders are shallower and we all adopt the setting $L = 6, H = 768, A = 12$.

We trained our models with 4 Nvidia Tesla V100 GPUs and optimized them using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.999$. Since the encoder is partly pre-trained (initialized with

[3] scripts available at https://github.com/nlpyang/PreSumm

[4] https://shorturl.at/nAG49

[5] We also tried RoBERTa$_{\text{LARGE}}$ and obtained worse results.

RoBERTa) and the decoders are initialized randomly, we set a lager learning rate for decoders. Specifically, we used 4e-5 for the encoder and 4e-4 for the decoders. Since CNN/DM is larger than NYT, we employed a batch size of 512 for CNN/DM and 64 for NYT (to ensure a sufficient number of model updates)[6]. Limited by the positional embedding of RoBERTa, all documents are truncated to 512 subword tokens. We trained our models on both CNN/DM and NYT for 100 epochs. It takes around one hour training on the CNN/DM and 30 minutes on the NYT for each epoch. The best checkpoint is at around epoch 85 on CNN/DM and epoch 65 on NYT according to validation sets.

When extracting the summary for a new document during test time, we rank all sentences using Equation (14) and select the top-3 sentences as the summary. When selecting sentences on the CNN/DM dataset, we find that *trigram blocking* (i.e., removing sentences with repeating trigrams to existing summary sentences) (Paulus et al., 2018) can reduce the redundancy, while *trigram blocking* does not help on NYT.

4.3 Results

Our main results are shown in Table 1. The first block includes several recent supervised models for document summarization. REFRESH (Narayan et al., 2018) is an extractive model, which is trained by globally optimizing the ROUGE metric with reinforcement learning. PTR-GEN (See et al., 2017) is a sequence to sequence based abstractive model with copy and coverage mechanism. Liu and Lapata (2019) initialize encoders of extractive model (BertSumExt) and abstractive model (BertSumAbs) with pre-trained BERT.

We present the results of previous unsupervised methods in the second block. LEAD-3 simply selects the first three sentences as the summary for each document. TEXTRANK (Mihalcea and Tarau, 2004) views a document as a graph with sentences as nodes and edge weights using the sentence similarities. It selects top sentences as summary w.r.t. PageRank (Page et al., 1999) scores. PACSUM (Zheng and Lapata, 2019) is yet another graph-based extractive model using BERT as sentence features. Sentences are ranked using centralities (sum of all out edge weights). They made the ranking criterion positional sensitive by forcing negative edge weights for edges between the current sentence and its preceding sentences. Adv-RF (Wang and Lee, 2018) and TED (Yang et al., 2020) are all based on unsupervised seq2seq auto-encoding with additional objectives of adversarial training, reinforcement learning and seq2seq pre-training to predict leading sentences.

PACSUM is based on the BERT (Devlin et al., 2019) initialization. RoBERTa (Liu et al., 2019), which extends BERT with better training strategies and more training data, outperforms BERT on many tasks. We therefore re-implemented PACSUM and extended it with both BERT and RoBERTa initialization (i.e., PACSUM (BERT) and PACSUM (RoBERTa))[7]. On CNN/DM, our re-implementation PACSUM (BERT) is comparable with Zheng and Lapata (2019). The results of PACSUM (BERT) and the RoBERTa initialized PACSUM (RoBERTa) are almost the same. Perhaps because it relies more on position information rather than sentence similarities computed by BERT or RoBERTa. STAS outperforms all unsupervised models in comparison on CNN/DM and the difference between STAS and all other unsupervised models are significant with a 0.95 confidence interval according to the ROUGE script. In the following, all significant tests on ROUGE are measured with a 0.95 confidence interval using the ROUGE script. Since STAS does not model sentence positions explicitly during ranking, while PACSUM does, we linearly combine the ranking scores of STAS and PACSUM (i.e., STAS + PACSUM)[8]. The combination further improves the performance.

On NYT, the trend is similar. STAS is slightly better than PACSUM although not significantly better (STAS is siginificantly better than all the other unsupervised models in comparison). Interestingly, there are also no significant differences between STAS and two supervised models (REFRESH and PTR-GEN). STAS + PACSUM even significantly outperforms the supervised REFRESH. The significant tests above all utilize the ROUGE script.

Examples of gold summaries and system outputs of REFRESH (Narayan et al., 2018), STAS and PACSUM (Zheng and Lapata, 2019) on the

[6]We used gradient accumulation technique (Ott et al., 2019) to increase the actual batch size.

[7]We re-implemented PACSUM, because the training code of PACSUM is not available.

[8]We first normalize sentence scores in each document for both STAS and PACSUM. In the combination, weight for STAS is 0.9 and weight for PACSUM is 0.1 (tuned on validation sets).

Method	CNN/DM			NYT		
	R-1	R-2	R-L	R-1	R-2	R-L
REFRESH (Narayan et al., 2018)	41.30	18.40	37.50	41.30	22.00	37.80
PTR-GEN (See et al., 2017)	39.50	17.30	36.40	**42.70**	**22.10**	**38.00**
BertSumExt (Liu and Lapata, 2019)	**43.25**	**20.24**	**39.63**	–	–	–
BertSumAbs (Liu and Lapata, 2019)	41.72	19.39	38.76	–	–	–
LEAD-3	40.50	17.70	36.70	35.50	17.20	32.00
TEXTRANK (tf-idf)	33.20	11.80	29.60	33.20	13.10	29.00
TEXTRANK (skip-thought)	31.40	10.20	28.20	30.10	9.60	26.10
TEXTRANK (BERT)	30.80	9.60	27.40	29.70	9.00	25.30
PACSUM (Zheng and Lapata, 2019)	40.70	17.80	36.90	41.40	21.70	37.50
PACSUM (BERT) *	40.69	17.82	36.91	40.67	21.09	36.76
PACSUM (RoBERTa) *	40.74	17.82	36.96	40.84	21.28	37.03
Adv-RF (Wang and Lee, 2018)	35.51	9.38	20.98	–	–	–
TED (Yang et al., 2020)	38.73	16.84	35.40	37.78	17.63	34.33
STAS	**40.90**	**18.02**	**37.21**	**41.46**	**21.80**	**37.57**
STAS + PACSUM	**41.26**	**18.18**	**37.48**	**42.42**	**22.66**	**38.50**

Table 1: Results on CNN/DM and NYT test sets using ROUGE F1. * means our own re-implementation. Results of TEXTRANK (tf-idf), TEXTRANK (skip-thought) and TEXTRANK (BERT) are reported in (Zheng and Lapata, 2019) and they use tf-idf, skip-thought vectors (Kiros et al., 2015) and BERT as sentence features, respectively.

Settings	valid set			test set		
	R-1	R-2	R-L	R-1	R-2	R-L
MSP	41.61	18.30	37.92	40.76	17.78	37.03
MSP+SS (STAS)	41.67	18.47	38.00	40.90	18.02	37.21
$\widetilde{r} = 1/\lvert\mathcal{D}\rvert$	41.58	18.43	37.89	40.74	17.88	37.04
$r' = 0$	33.92	12.93	30.99	33.30	12.61	30.33

Table 2: Ablation study on CNN/DM validation and test sets using ROUGE F1.

	valid set			test set		
	R-1	R-2	R-L	R-1	R-2	R-L
w/ $A_{i,j}$	33.66	12.78	30.75	33.02	12.48	30.08
w/ $A_{j,i}$	41.67	18.47	38.00	40.90	18.02	37.21

Table 3: $\mathbf{A_{j,i}}$ v.s. $\mathbf{A_{i,j}}$ on CNN/DM validation and test sets using ROUGE F1.

CNN/DM dataset can be found in appendix B.

4.4 Analysis

Ablation Study In Section 3.2, we proposed two pre-training tasks. *Are they are all useful for summarization?* As shown in Table 2, when we only employ the masked sentences prediction task (MSP), we can obtain a ROUGE-2 of 17.73, which is already very close to the result of PACSUM (see Table 1). When we add the sentence shuffling task (denoted as MSP+SS (STAS)), we improves the performance over MSP. Note that we can not use only the sentence shuffling task (SS), because the first term in our sentence scoring equation (see Equation (14)) depends on the probabilities produced by decoder in the MSP task.

In section 3.3, we propose two criteria to score sentences (see the two terms in Equation (14)). The effects of them are shown in the second block of Table 2. Since the attention based criterion r' relies on sentence probability based criterion $\widetilde{r}$, we cannot remove $\widetilde{r}$ and instead we set $\widetilde{r} = \frac{1}{\mathcal{D}}$ to see the effect of $\widetilde{r}$. As a result, ROUGE-2 decreases by 0.14, which indicates that $\widetilde{r}$ is necessary for ranking. Also note that when setting $\widetilde{r} = \frac{1}{\mathcal{D}}$, our method is equivalent to PageRank using sentence level attention scores as edge weights. Instead of iterating until convergence as in the original PageRank algorithm, we find a small iteration number ($T \leq 3$) is sufficient. To study the effect of the attention based criterion r', we set $r' = 0$, which means sentences are ranked using sentence probability based criterion $\widetilde{r}$. We can see that the performance drops dramatically by 5 ROUGE-2.

$\mathbf{A_{j,i}}$ v.s. $\mathbf{A_{i,j}}$ In Equation (13), we compute r'_i with $\mathbf{A}_{j,i}$ (attention score from S_j to S_i). The intuition of using $\mathbf{A}_{j,i}$ is that a sentence is important if the interpretation of other important sentences depends on it. However, an alternative is to use $\mathbf{A}_{i,j}$. It shows in Table 3 that $\mathbf{A}_{j,i}$ is indeed better.

Sentence Position Distribution We also analyze *how extractive sentences by different models are distributed in documents?* We compare STAS against LEAD-3, PACSUM and ORACLE using the first 12 sentences in all documents on CNN/DM test set. ORACLE is the upper bound for extractive models. Extractive summaries of ORACLE are generated by selecting a subset of sentences in a document, which maximize ROUGE score (Nallapati et al., 2017). As shown in Figure 4, we can see

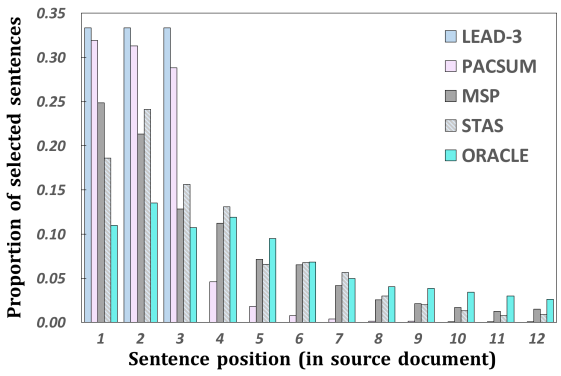

Figure 4: Proportion of extracted sentences by different unsupervised models against their positions.

that sentences selected by ORACLE are smoothly distributed across all positions, while LEAD-3 only selects the first 3 sentences. Compared to STAS, the sentence distribution of PACSUM is closer to that of LEAD-3 and STAS produces a sentence distribution that is more similar to that of ORACLE. The observation above indicates that our model relies less on sentence positions compared to PACSUM. We further computed the Kullback Leibler divergence between the sentence position distribution of an unsupervised model and the distribution of ORACLE and we denote it as $KL(\cdot||\text{ORC})$. We found $KL(\text{PACSUM}||\text{ORC}) = 0.614$ is much large than $KL(\text{STAS}||\text{ORC}) = 0.098$, indicating STAS is better correlated with ORACLE. We introduce the sentence shuffling task to encourage STAS to select sentences based on their contents rather than their positions only (see Section 3.2). After we remove the sentence shuffling task from STAS during pre-training (see MSP in Figure 4), there is a clear trend that leading sentences are more frequently selected. Moreover, $KL(\text{STAS}||\text{ORC}) < KL(\text{MSP}||\text{ORC}) = 0.108$. By introducing the sentence shuffle task, sentence positional distribution of STAS is closer to that of ORACLE.

	valid set			test set		
	R-1	R-2	R-L	R-1	R-2	R-L
MSP	41.61	18.30	37.92	40.76	17.78	37.03
MSP+(a)	40.93	17.73	37.27	40.15	17.25	36.46
MSP+(b)	37.59	14.71	33.95	36.91	14.26	33.25
MSP+SS	41.67	18.47	38.00	40.90	18.02	37.21

Table 4: Compare SS with other two different methods which remove sentence positions. (a) remove the sentence level positional embedding; (b) for each sentence in the token level, use a positional embedding from positional 0. Results are reported on CNN/DM.

Why Sentence Shuffling? Since the Sentences Shuffling task aims to make STAS less dependent on sentence positions. However, there are potentially simpler methods to remove sentence position information. For example, (a) we can remove the sentence-level positional embedding and (b) for each sentence in the token level, we can use a positional embedding from positional 0. Results in Table 4 indicates that upon the MSP objective, strategies (a) and (b) hurt the performance of MSP significant, while SS improves over MSP. It may because positional embeddings, whether on token or sentence level, are important (at least for the MSP task). One advantage of SS over (a) and (b) is that it can make our model less dependent on positions (see Section 4.4) and retain the power of positional embeddings and the MSP objective at the same time.

5 Conclusions

In this paper, we find that (sentence-level) transformer attention (in a hierarchical transformer) can be used to rank sentences for unsupervised extractive summarization, while previous work leverage graph based (or rule based) methods and sentence similarities computed with off-the-shelf sentence embeddings. We propose the sentence shuffling task for pre-training hierarchical transformers, which helps our model to select sentences based on their contents rather than their positions only. Experimental results on CNN/DM and NYT datasets show that our model outperforms other recently proposed unsupervised methods. The sentence position distribution analysis shows that our method is less dependent on sentence positions. When combined with recent unsupervised model explicitly modeling sentence positions, we obtain even better results. In the next step, we plan to apply our models to unsupervised abstractive summarization.

Acknowledgments

We gratefully acknowledge Hao Zheng for the technical advice during our re-implementation of PACSUM (Zheng and Lapata, 2019). We would also like to thank the anonymous reviewers for their insightful feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly

learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. SEQˆ3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. 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 673–681, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336.

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. Association for Computational Linguistics.

Eric Chu and Peter J. Liu. 2019. Meansum: A neural model for unsupervised multi-document abstractive summarization. In *ICML*.

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.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, pages 13042–13054.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Text Summarization Branches Out*, pages 104–111, Barcelona, Spain. Association for Computational Linguistics.

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, Brussels, Belgium. Association for Computational Linguistics.

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, Berlin, Germany. 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.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA. Association for Computational Linguistics.

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. 2020. Pretraining with contrastive sentence objectives improves discourse performance of language models. *arXiv preprint arXiv:2005.10389*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

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.

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. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2002. From single to multi-document summarization. In *Proceedings of the 40th annual meeting of the association for computational linguistics*, pages 457–464.

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 3730–3740, Hong Kong, China. Association for Computational Linguistics.

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*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

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. Association for Computational Linguistics.

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.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1997. Pagerank: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954, Lisbon, Portugal. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D 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, New Orleans, Louisiana. Association for Computational Linguistics.

Dragomir R Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization.(2004). *LREC, Lisbon, Portugal*.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

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/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

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. 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.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Honolulu, Hawaii. Association for Computational Linguistics.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306.

Yaushian Wang and Hung-Yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium. Association for Computational Linguistics.

Zi-Yi Yang, Chenguang Zhu, Robert Gmyr, Michael Zeng, and Xuedong Huang. 2020. Ted: A pretrained unsupervised summarization model with theme modeling and denoising. *ArXiv*, abs/2001.00725.

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, Brussels, Belgium. Association for Computational Linguistics.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2019. Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, Florence, Italy. Association for Computational Linguistics.