Stepwise Extractive Summarization and Planning with Structured Transformers

Shashi Narayan* **Joshua Maynez*** **Jakub Adamek**
{shashinarayan,joshuahm,enkait}@google.com

Daniele Pighin **Blaž Bratanič** **Ryan McDonald**
{biondo,blazb,ryanmcd}@google.com

Google Research

Abstract

We propose encoder-centric stepwise models for extractive summarization using structured transformers – HiBERT (Zhang et al., 2019) and Extended Transformers (Ainslie et al., 2020). We enable stepwise summarization by injecting the previously generated summary into the structured transformer as an auxiliary sub-structure. Our models are not only efficient in modeling the structure of long inputs, but they also do not rely on task-specific redundancy-aware modeling, making them a general purpose extractive content planner for different tasks. When evaluated on CNN/DailyMail extractive summarization, stepwise models achieve state-of-the-art performance in terms of Rouge without any redundancy aware modeling or sentence filtering. This also holds true for Rotowire table-to-text generation, where our models surpass previously reported metrics for content selection, planning and ordering, highlighting the strength of stepwise modeling. Amongst the two structured transformers we test, stepwise Extended Transformers provides the best performance across both datasets and sets a new standard for these challenges.[1]

1 Introduction

Extractive document summarization is the task of creating a summary by identifying (and subsequently concatenating) the most important sentences in a document (Erkan and Radev, 2004; Nenkova and McKeown, 2011). In recent years this task has matured significantly, mostly thanks to advances in deep neural networks. Cheng and Lapata (2016) conceptualize extractive summarization as a sequence labeling task in which first a hierarchical long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) is used to encode a document and then another LSTM is used to predict for each sentence whether it should be included in the summary. This architecture was later adopted by Nallapati et al. (2016a), Nallapati et al. (2017), Narayan et al. (2018b), Zhang et al. (2018) and Dong et al. (2018).

Following the success of pre-trained transformer-based architectures for many tasks (Vaswani et al., 2017; Devlin et al., 2019), the current state-of-the-art approach to extractive summarization uses transformers to learn sentence representations and to rank sentences by their saliency (Liu, 2019; Liu and Lapata, 2019b; Zhang et al., 2019; Zhong et al., 2019a; Bi et al., 2020). The top scoring sentences are then assembled to produce an extract of the document. Summaries built in this fashion (Cheng and Lapata, 2016; Narayan et al., 2018a; Zhang et al., 2018; Dong et al., 2018) are prone to contain redundant information. Several recent approaches have explored mechanisms to better handle redundancy, such as heuristic-based Trigram Blocking (TriBlk; Liu and Lapata, 2019b; Wang et al., 2020), hand-crafted feature-driven models (Ren et al., 2017) and redundancy aware neural sequence models (Zhou et al., 2018; Bi et al., 2020). One common problem with these models is that their focus is limited to content overlap and to respecting length budgets. However, these are but a small subset of the dimensions necessary to produce informative and coherent summaries. Ideally, models would utilize enriched document and summary representations in order to implicitly learn better extractive plans for producing summaries (Liu et al., 2019a; Mendes et al., 2019). One such method is *stepwise* summarization (Liu et al., 2019a), where a summary is constructed incrementally by choosing new content conditioned on previously planned content.

In this paper, we propose encoder-centric stepwise models for extractive summarization using

* Equal contribution.

[1] The code and data are available at https://github.com/google-research/google-research/tree/master/etcsum.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4143–4159,
November 16–20, 2020.

structured transformers. Structured transformers are transformer-based architectures that have the flexibility to model some form of structure of the input, e.g., hierarchical document structure. In this paper, we specifically study two such architectures – HiBERT (Zhang et al., 2019) and Extended Transformers Construction (ETC; Ainslie et al., 2020). Details of these are given in Sections 4 and 5. We enable stepwise summarization by injecting the previously planned summary content into the structured transformer as an auxiliary sub-structure. The model then can holistically learn any document-level coherence properties, such as saliency, redundancy, and ordering, embodied in the gold summaries. This differs from other methods which are either task specific (e.g., redundancy aware modeling in Bi et al., 2020) or not holistic (e.g., manually curated features in Liu et al., 2019a). An added advantage of structured encoders is that they break the quadratic attention mechanism of transformers (Devlin et al., 2019), making them more efficient and able to process longer inputs, instead of truncating the inputs to 512 tokens (Liu and Lapata, 2019b; Bi et al., 2020), which is critical for long inputs and outputs which require non-trivial planning. When evaluated on the CNN/DailyMail summarization dataset (Hermann et al., 2015), we achieve state-of-the-art performance in terms of Rouge (Lin and Hovy, 2003) without any redundancy (Zhou et al., 2018; Bi et al., 2020) or sentence selection mechanisms (Liu and Lapata, 2019b).

Our model's task-agnostic approach allows it to implicitly learn and leverage content plans directly from the data. Moreover, structured transformers form the basis of our model, which are flexible in terms of content type (e.g., text or tables) that can be modeled. We demonstrate this by learning intricate extractive content plan for the Rotowire table-to-text generation task (Wiseman et al., 2017). This task requires the generation of long summaries from large score tables detailing the the specifics of a sports match, which often necessitates dedicated content selection and planning models to generate a high-quality summary (Wiseman et al., 2017; Puduppully et al., 2019a). We show that our stepwise framework achieves higher content selection, planning and ordering scores relative to prior work with task-specific planning mechanisms.

The contributions of the paper are as follows: 1) this is first study to use ETC (Ainslie et al., 2020) for summarization for its ability and flexibility to better model long and structured inputs; 2) we propose augmentions of two structured transformers, HiBERT and ETC, in order to enable stepwise models for extractive planning; 3) we demonstrate empirically that our models are general purpose and can be adapted as an extractive document summarizer or as a content planner for table-to-text generation; 4) Our experiments highlight the effectiveness of stepwise modeling, specifically stepwise ETC, which sets a new standard for both tasks.

2 Related Work

Redundancy. Summarization models often use a dedicated *sentence selection* step after *sentence scoring* to address redundancy. Maximal Marginal Relevance (Carbonell and Goldstein, 1998) based methods select the content that has the maximal score and is minimally redundant with the previously constructed partial summary. Others treated sentence selection as an optimization problem under some constraints such as summary length (McDonald, 2007; Lin and Bilmes, 2011). Liu and Lapata (2019b) and Wang et al. (2020) used heuristic-based Trigram Blocking (TriBlk) for redundancy elimination. Ren et al. (2017) trained two neural networks with handcrafted features; one is used to rank sentences, and the other one is used to model redundancy during sentence selection. Zhou et al. (2018) and Bi et al. (2020) proposed redundancy-aware models by modeling redundancy and saliency jointly during the scoring process using neural sequence models. In contrast to these approaches, our models are not redundancy-aware. Instead, they implicitly model redundancy by injecting previously generated summary representations. By virtue of this our models are not text-specific and can be applied to other tasks (see Section 7).

Partial Summary Representations. Ultilizing representations of partially generated summaries is relatively less studied in summarization. Mendes et al. (2019) proposed to dynamically model the generated summary using an LSTM to iteratively increment summaries based on previously extracted information. Liu et al. (2019a) used a feed-forward neural network driven by hand-curated features capturing the prevalence of domain subtopics in the source and the summary. To the best of our knowledge, our models are first to use summary representations with structured transformers for summarization. Our models learn to make summary-informed next-sentence predictions without any hand-curated features.

Long-form Summarization. It is well known that a better content selection benefits abstractive summarizers to generate summaries that are not only fluent but also informative (Gehrmann et al., 2018; Hsu et al., 2018; Xiao et al., 2020). It can be particularly important when generating long abstractive summaries (Liu et al., 2018; Liu and Lapata, 2019a) or summarizing multiple documents (Yasunaga et al., 2017). Earlier multi-document summarization methods have addressed the issue of long form input by graph-based representations of sentences or passages (Erkan and Radev, 2004; Christensen et al., 2013). Recently, Yasunaga et al. (2017) proposed a neural version of this framework using graph convolutional networks (Kipf and Welling, 2017). Liu and Lapata (2019a) used cross-document attention mechanism to share information as opposed to simply concatenating text spans using hierarchical transformers. Similar to this motivation, we also explore better encoding of long inputs with structured transformers.

Table-to-Text Content Planning. Wiseman et al. (2017) introduced the Rotowire dataset, which requires multi-sentence summaries of large tables. Several works found that the key to generate fluent and informative summaries for this task is to have dedicated content planning and realization steps (Puduppully et al., 2019a,c; Miculicich et al., 2019). Miculicich et al. (2019) and Gong et al. (2019b) used a transformer encoder, and, Gong et al. (2019a) used multi-dimensional hierarchical LSTM encoders to compute better table entry representations. Following these lines of work, we evaluate our models to generate long content plans for this task using structured transformers.

3 Problem: Stepwise Content Extraction

We define a general paradigm for stepwise content extraction that can be easily tailored to both extractive summarization and table-to-text generation. Given an input $D = \{s_1, s_2, \ldots, s_n\}$ with n content units, the goal is to learn an extractive content plan, i.e., $S'_m = \{s'_j | 1 \leq j \leq m, s'_j \in (D \cup \{\emptyset\})\}$, of length m; s'_m is an empty unit ($\emptyset$) denoting the end of the plan. We formulate this as an iterative ranking problem (Liu et al., 2019a; Bi et al., 2020) where at each k-th step ($1 \leq k \leq m$) given the input D and the previously selected plan S'_{k-1}, we select $s'_k \in (D \cup \{\emptyset\})$ with a probability $p(s'_k | S'_{k-1}, D; \theta)$ with model parameters θ. The selected content is then added to S'_{k-1} to construct S'_k. The best plan $\hat{S}$ can be defined as:

$$\hat{S} = \arg\max_{S'_m, \forall m} \prod_{k=1}^{m} P(s'_k | S'_{k-1}, D; \theta).$$

For extractive document summarization, let $D = \{s_1, s_2, \ldots, s_n\}$ be a document with n sentences. Our goal is to learn an extractive plan (or summary in this case) $\hat{S}$ which best summarizes D. For table-to-text generation, we represent a table with n records as $D = \{s_1, s_2, \ldots, s_n\}$. We aim to generate a plan S'_m that can be used by a text generator to generate a meaningful and coherent summary.

For exposition, we use the extractive document summarization setup to introduce our stepwise models with HiBERT (Zhang et al., 2019) and ETC (Ainslie et al., 2020) in the following sections. Specifically, we use 'sentence' as a content unit and 'previously' or 'partially generated summary' for a previously selected content plan.

4 Stepwise HiBERT

Hierarchical encodings have been used to model input structure with LSTMs (Nallapati et al., 2016b; Cheng and Lapata, 2016; Narayan et al., 2018b). Zhang et al. (2019) proposed **HiBERT** with two stacked Transformer encoders (Vaswani et al., 2017) for extractive summarization (see the middle diagram in Figure 1): a *sentence encoder* that *independently* builds representations for each sentence in the document; and a *document encoder* that operates over sentence encodings to build contextual representations for all sentences. These contextual sentence representations are then ingested by a classifier to predict the salience score of each sentence in the document. As in standard transformers, both encoders have multiple layers with each layer composed of a multi-head self-attention layer followed by a feed-forward sub-layer with residual connections (He et al., 2015) and layer normalizations (Ba et al., 2016). For **Stepwise HiBERT**, at time step k, we modify the document encoder with the content plan S'_{k-1}, which is the previously selected sentences in the summary. This is depicted in Figure 2 (left) and allows the model to implicitly select new sentences relative to the previously generated summary.

Sentence and Document Encoders. Let $D = \{s_1, s_2, \ldots, s_n\}$ be a document, where $s_i = \{w_1^i, w_2^i, \ldots, w_{|s_i|}^i\}$ is a sentence in D and w_j^i is a token in s_i. s_i is first mapped to a continuous space $\mathbf{E}_{s_i} = \{\mathbf{e}_1^i, \mathbf{e}_2^i, \ldots, \mathbf{e}_{|s_i|}^i\}$ where $\mathbf{e}_j^i = \mathbf{e}(w_j^i) + \mathbf{p}_j^{\text{token}}$. $\mathbf{e}(w_j^i)$ and $\mathbf{p}_j^{\text{token}}$ are the token

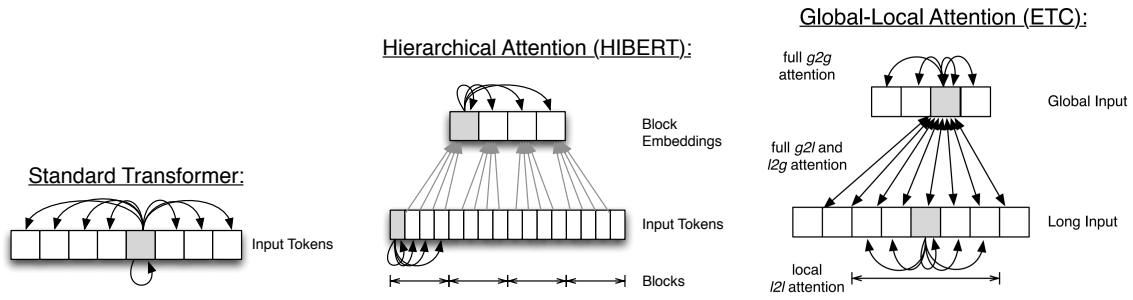

Figure 1: Memory usage and attentions in standard transformers (Devlin et al., 2019), HiBERT (Zhang et al., 2019) and ETC (Ainslie et al., 2020).

and positional embeddings of token w_j^i, respectively. Our Transformer-based sentence encoder then transforms $\mathbf{E}_{s_i}$ into a list of hidden representations $\{\mathbf{h}_1^i, \mathbf{h}_2^i, \ldots, \mathbf{h}_{|s_i|}^i\}$, where $\mathbf{h}_j^i$ is the hidden representation for w_j^i. Following the standard practice (Devlin et al., 2019; Liu and Lapata, 2019b), we take the first hidden representation $\mathbf{h}_1^i$ as the representation for the sentence s_i.

Zhang et al. (2019) use a standard Transformer document encoder. It takes the document representation $\hat{\mathbf{H}}_D = \{\hat{\mathbf{h}}^1, \hat{\mathbf{h}}^2, \ldots, \hat{\mathbf{h}}^n\}$, where $\hat{\mathbf{h}}^i = \mathbf{h}_1^i + \mathbf{p}_i^{\text{sent}}$. $\mathbf{h}_1^i$ and $\mathbf{p}_i^{\text{sent}}$ are the representation from the sentence encoder and the positional embedding for sentence s_i in the document, respectively, and, builds contextual sentence representations $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n\}$.

Stepwise Modeling. At step k, let $S'_{k-1} = \{s'_1, s'_2, \ldots, s'_{k-1}\}$ be the partial summary with $(k-1)$ previously extracted sentences. In addition to $\hat{\mathbf{H}}_D$, our document encoder takes the summary representation $\hat{\mathbf{H}}_{S'_{k-1}} = \{\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^{k-1}\}$, where $\hat{\mathbf{x}}^i = \mathbf{h}_1^i + \mathbf{p}_i^{\text{sum}}$. $\mathbf{h}_1^i$ is the representation from the sentence encoder for sentence s_i and $\mathbf{p}_i^{\text{sum}}$ is the positional embedding for sentence s_i in S'_{k-1}. At each layer, the document encoder employs three levels of nested multi-headed attentions (Vaswani et al., 2017) to build summary-informed contextual sentence representations $\{\mathbf{d'}_1, \mathbf{d'}_2, \ldots, \mathbf{d'}_n\}$: *document self-attention*, *summary self-attention* and *document-summary attention* (see Figure 2, left). The first two operate in parallel, followed by the document-summary attention.

While document self-attention learns the contextual hidden representation $\mathbf{h}_{s_i}^{\text{doc}\to\text{doc}}$ of each sentence in the document D, summary self-attention learns the contextual hidden representation $\mathbf{h}_{s'_i}^{\text{sum}\to\text{sum}}$ of each sentence in S'_{k-1}. We share the parameters of the document and summary self-attention layers. The document-summary attention then builds the contextual hidden representation $\mathbf{h}_{s_i}^{\text{doc}\to\text{sum}}$ of each sentence in the document D using linear projections of $\mathbf{h}_{s_i}^{\text{doc}\to\text{doc}}$ as query, and $\mathbf{h}_{s'_i}^{\text{sum}\to\text{sum}}$ as key and values (Vaswani et al., 2017).

In addition to the introduction of stepwise mechanism to HiBERT, our positional embeddings, $\mathbf{p}_j^{\text{token}}$, $\mathbf{p}_j^{\text{doc}}$ and $\mathbf{p}_j^{\text{sum}}$, are not shared to better model individual sentences, the document and the different styles of summary. Zhang et al. (2019) shared their token ($\mathbf{p}_j^{\text{token}}$) and sentence ($\mathbf{p}_j^{\text{sent}}$) positional embeddings. But we both use the absolute position encodings used in the original BERT model (Devlin et al., 2019).

5 Stepwise ETCSum

There has been growing interest in addressing the limitation of the transformer architecture used in BERT (Devlin et al., 2019) where memory usage scales quadratically with the size of the input (Guo et al., 2019; Dai et al., 2019; Ye et al., 2019; Child et al., 2019; Rae et al., 2020; Beltagy et al., 2020; Roy et al., 2020). HiBERT alleviates this problem by modeling each sentence independently; the memory usage in HiBERT scales with the square of the number of sentences, and the square of the maximum length of any sentence. However, the main disadvantage of this approach is that token-level attention across sentences is prohibited and long range attention only happens indirectly at the second-stage encoder (see the middle diagram in Figure 1). Recently, Extended Transformer Construction (ETC; Ainslie et al., 2020) provides an alternative. It alleviates the quadratic memory growth by introducing sparsity to the attention mechanism via its novel global-local attention mechanism (see the rightmost diagram in Figure 1). This not only permits encoding of long inputs,[2] but also enables a mechanism to model structure directly through nodes in the global attention layer.

[2] As do other recent architectures (Yang et al., 2019; Kitaev et al., 2020).

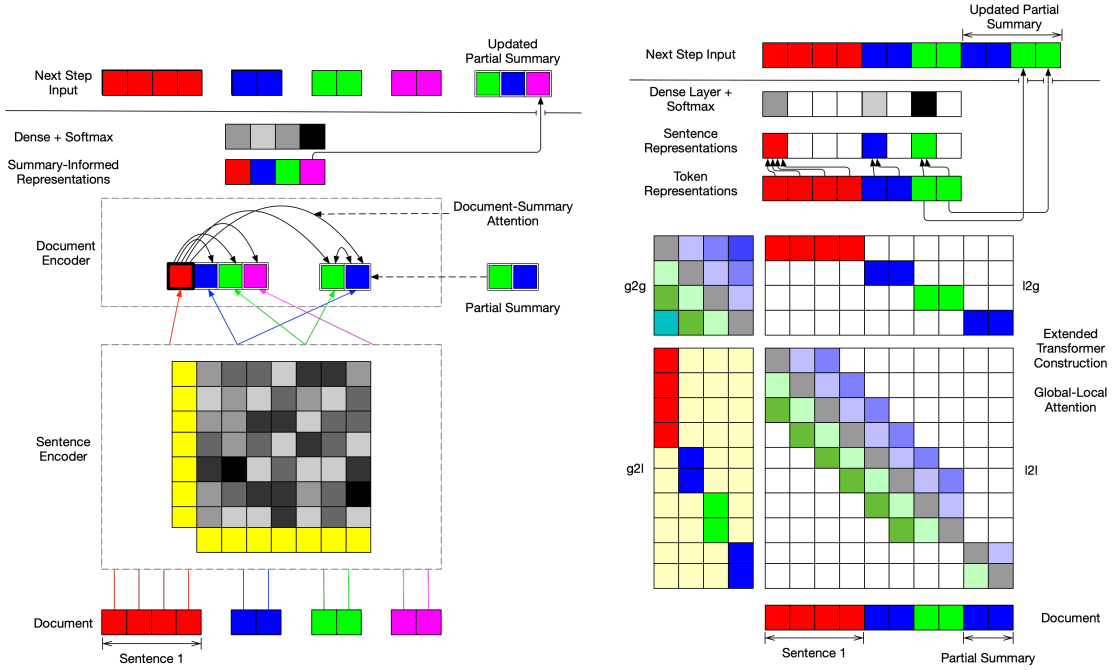

Figure 2: Stepwise HiBERT (left) and ETCSum (right) models. HiBERT builds summary informed representation by jointly modeling partially generated summary and the document during document encoding, while ETCSum takes as input the document appended with the partially generated summary.

Global-Local Attention. The ETC model architecture receives two inputs: a long input, which in most cases corresponds to the text to be encoded; and an auxiliary global input, which serves as inductive bias features. First, the model builds an attention map, called *long-to-long*, across the long input with a sparse local attention of fixed length, this bypasses the quadratic memory complexity and allows to scale input lengths to the thousands of tokens, but limits the attention span of tokens to their nearest neighbors.

To overcome this limitation, the global-local attention defines three other attention parts: *global-to-global*, *global-to-long* and *long-to-global*, all with unrestricted attention. This allows tokens arbitrarily far apart to attend to each other with at most one hop through the global input tokens. We refer the reader to Ainslie et al. (2020) for more details. The right parts of Figures 1 and 2 illustrate these four types of attentions and the sparsity diagrams where each cell in a row i and column j is different than white input token w_i can attend to input token w_j, same relative position embeddings are indicated by using the same color.

Stepwise Modeling. Given the document D and its partial summary S'_{k-1} at step k, we construct an input $I = D^\frown S'_{k-1} = \{w_1, \ldots, w_{|D^\frown S'_{k-1}|}\}$ by concatenating the document D and the partial summary S'_{k-1}. ETC replaces absolute position encodings with relative position encodings (Shaw et al., 2018) to easily adapt to greater input lengths than seen during pretraining. In addition to modeling relative positions in an input sequence, relative position encodings in ETC are also used to model arbitrary pairwise token relations useful for structured inputs.

We used the auxiliary global input to represent sentence structure. Specifically, following (Ainslie et al., 2020), we placed one auxiliary token in the global input per each sentence in the input I. We linked the global tokens with the input tokens by using relative position labels to represent whether each token belongs to that sentence. Global-to-global attention is left unrestricted, allowing all sentences to attend to each other. This result is summary-informed contextualized input token representations via attention through the global nodes. In the rest of the paper we refer to this summarizer by **Stepwise ETCSum**. Similar to HiBERT, we take the first token hidden representation $\mathbf{h}_1^i$ as the representation for the sentence s_i. Finally, sentence embeddings are passed to the softmax layer for salience scoring. Both HiBERT and ETCSum are then trained with the cross entropy loss.

6 Extractive Document Summarization

6.1 Experimental Setup

Dataset. We evaluate our models on the CNN and DailyMail news highlights datasets (Hermann et al., 2015). We used standard splits

(287,227/13,368/11,490 documents) for training, validation, and testing. We did not anonymize entities or lower case tokens as in (Narayan et al., 2018b; Zhou et al., 2018; Zhang et al., 2019; Liu and Lapata, 2019b). The documents in the CNN/DailyMail dataset are long; the average lengths are 760.5 words (34 sentences) for CNN and 653.3 words (29.3 sentences), for DailyMail. The human written abstracts have 46 and 55 words for CNN and DailyMail, respectively. We evaluated summarization quality using F_1 Rouge.[3]

Baselines. We compared our Stepwise HiBERT and ETCSum models to Lead and Oracle baselines. Lead selects the first 3 sentences to form the summary, while Oracle baselines creates a summary by selecting the best possible set of sentences in the document that gives the highest average of Rouge-1, Rouge-2 and Rouge-L F1 scores with respect to the human written summary. The Oracle (512) truncates the input document to 512 tokens. We further compared our models against several redundancy-aware models (NeuSum; Zhou et al., 2018 and ARedSum; Bi et al., 2020) and models that uses Trigram Blocking (TriBlk; Liu and Lapata, 2019b) for redundancy elimination during sentence selection (see the second block in Table 1).

To understand the importance of modeling long documents for extractive summarization, we also trained BERTSum, similar to Liu and Lapata (2019b), with a receptive capacity of 512 tokens initialized with the BERT checkpoint. Our BERTSum differs slightly from Liu and Lapata (2019b), in that we don't use segment embeddings. We also report on RoBERTa (Liu et al., 2019b) initialized version of BERTSum (RoBERTaSum).

We also trained non-stepwise variants of HiBERT and ETCSum models (the third block in Table 1). In this setting, HiBERT and ETC do not take partial summaries as input. Instead, they simply take the input document and generate salient scores (using a sigmoid layer) for each sentence in the document; the top three sentences are then assembled to generate the summary. Our implementation of HiBERT differs from Zhang et al. (2019). For example, we don't pretrain HiBERT from scratch for document modeling as in Zhang et al. (2019). Instead, we initialize our HiBERT models with publicly available RoBERTa (Liu et al., 2019b) checkpoints following the superior performance of

Models	R1	R2	RL
Lead	40.42	17.62	36.67
Oracle (512)	52.59	31.24	48.87
Oracle (Full)	57.82	35.05	53.99
Latent (Zhang et al., 2018)	41.05	18.77	37.54
Refresh (Narayan et al., 2018b)	41.00	18.80	37.70
BanditSum (Dong et al., 2018)	41.50	18.70	37.60
NeuSUM (Zhou et al., 2018)	41.59	19.01	37.98
ExConSum (Mendes et al., 2019)	41.70	18.60	37.80
JECS (Xu and Durrett, 2019)	41.70	18.50	37.90
LSTM+PN (Zhong et al., 2019b)	41.85	18.93	38.13
HER (Luo et al., 2019)	42.30	18.90	37.60
HiBERT (Zhang et al., 2019)	42.37	19.95	38.83
PNBERT (Zhong et al., 2019a)	42.69	19.60	38.85
BERTSum (Liu and Lapata, 2019b)	42.61	19.99	39.09
BERTSum+TriBlk	43.25	20.24	39.63
ARedSum-CTX (Bi et al., 2020)	43.43	20.44	**39.83**
HSG (Wang et al., 2020)	42.31	19.51	38.74
HSG+TriBlk	42.95	19.76	39.23
BERTSum Large*	43.85	20.34	39.90
Our non-stepwise models			
BERTSum	41.55	19.34	37.80
BERTSum+TriBlk	42.70	19.93	38.89
RoBERTaSum	42.99	20.60	39.21
RoBERTaSum+TriBlk	43.30	20.58	39.48
HiBERT	41.43	19.23	37.73
HiBERT+TriBlk	42.37	19.68	38.63
ETCSum	42.67	20.27	38.90
ETCSum+TriBlk	43.43	20.54	39.58
Our stepwise models			
Stepwise RoBERTaSum	41.99	19.78	37.76
Stepwise RoBERTaSum+TriBlk	41.50	19.48	37.25
Stepwise HiBERT	41.98	19.53	38.32
Stepwise HiBERT+TriBlk	42.12	19.45	38.43
Stepwise ETCSum	**43.84**	**20.80**	39.77
Stepwise ETCSum+TriBlk	43.23	20.30	39.15

Table 1: Rouge F1 scores on the CNN/DailyMail test set. Boldfaced numbers are the best results among comparable models. * BERTSum Large builds on BERTLarge (24 layers) architectures, whereas ours build on BERTBase (12 layers) architectures.

RoBERTaSum over BERTSum. We use different number of layers in the document encoder (L_{doc} = 3) and in the sentence encoder (L_{sent} = 9), as opposed to equal number of layers (L = 6) in both encoders of Zhang et al. (2019). The layers in the document and sentence encoders were initialized with the top and the bottom layers of RoBERTa, respectively. All ETCSum models were initialized with the uncased version of ETC pretrained checkpoints (Ainslie et al., 2020) pretrained using the standard masked language model task and the contrastive predictive coding (van den Oord et al., 2018).[4]

We also report on the effect of TriBLK with all our models. We only experiment with the base-sized models and therefore have 12 layers, a hidden size of 768, filter size of 3072, and 12 attention

[3]We lowercased candidate and reference summaries and used `pyrouge` with parameters "-a -c 95 -m -n 4 -w 1.2."

[4]We thank the authors (Ainslie et al., 2020) for sharing their ETC checkpoints with us.

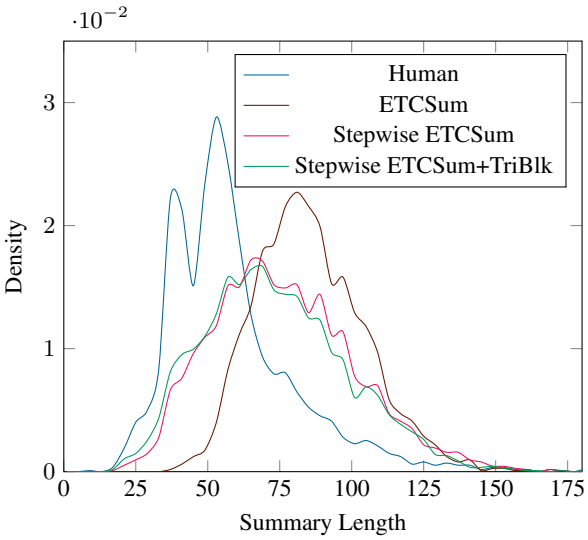

Figure 3: Length distributions in ETCSum summaries on the CNN/DailyMail test set.

heads. For comparison, we report results from BERTSum Large (Liu and Lapata, 2019b) which uses 24 layers. Finally, we employ a beam decoding to predict summaries using our stepwise models; we use a beam size of 3 for a maximum of 4 steps. We don't allow repeated sentences, though this is not a requirement. We refer the reader to the supplementary material for implementation and reproducibility details.

Generating Extractive Oracles. Following Narayan et al. (2018b), we train models to predict all sentences in Oracle (Full) for non-stepwise training. Stepwise training learns to do this gradually: at each step, we train model to predict the next sentence in Oracle (Full) using the earlier predicted sentences and the document. During testing, human written abstracts are used as reference summaries to evaluate our models.

6.2 Results

Long form Summarization. In our experiments, ETCSum appears to be far more superior than HiBERT when modeling long documents for extractive summarization; ETCSum outperformed HiBERT in all cases including stepwise or non-stepwise predictions, and, with or without trigram blocking. The downside of HiBERT where token-level attention across sentences is not possible, is not optimal for modeling documents. Both ETCSum and ETCSum+TriBlk performed better than BERTSum and BERTSum+TriBlk, respectively. These results suggest the importance of modeling the whole document with ETCSum, rather than truncating it to only 512 tokens to fit BERTSum. However, the improvement may not be attributed solely to ETCSum's ability to model long inputs, but also to its better initialization with ETC checkpoints (Ainslie et al., 2020), specially when the improvement diminishes when compared against RoBERTaSum.[5]

Stepwise vs Non-stepwise models. First of all, trigram filtering seems to be the key in addressing redundancy in generated summaries in non-stepwise models. It helps almost all models including our HiBERT and ETCSum (except for the single case of RoBERTaSum on Rouge-2). Interestingly, we don't observe the same pattern for our stepwise models. We observe that our stepwise models (both HiBERT and ETCSum, without TriBlk) consistently improve over their non-stepwise counterparts. But when stepwise is applied with TriBlk, we don't always see improvements. We conjecture that our stepwise models themselves are inherently better at avoiding redundancy in generated summaries due to the knowledge of previously generated summary at each prediction step, and improvements with TriBlk are not always complementary. The same is also demonstrated in Figure 3; density curves show that Stepwise ETCSum (avg:76.96, std:24.77) follows the human distribution (avg:58.3, std:24.8) better than ETCSum (avg:85.84, std:19.06). With Stepwise ETCSum+TriBlk (avg:73.92, std:24.76), we don't see significant improvement over Stepwise ETCSum.

We also report on Stepwise RoBERTaSum baselines and performance dropped compared to corresponding non-stepwise models. Perhaps without any structure in the transformer, simple summary concatenation is not a good method for Stepwise RoBERTaSum to distinguish the document from the summary. There might be better ways (than the vanilla concatenation), but with Stepwise ETCSum or HiBERT, it is very natural. Stepwise RoBERTaSum also loses access to the end of the input as the partial summary grows for documents that are already close to 512 tokens in length.

Finally, our Stepwise ETCSum model without any explicit redundancy or sentence selection mechanisms, achieved comparable performance to the state of the art on the CNN/DailyMail

[5]One may consider to access the modeling of long inputs in ETCSum against the truncated inputs in BERTSum and RoBERTaSum, by initializing ETCSum with BERT or RoBERTa checkpoints, and not ETC checkpoint. However, this is not fair to ETCSum as BERT or RoBERTa uses absolute position embeddings (Devlin et al., 2019), whereas, ETC uses relative position embeddings (Shaw et al., 2018).

Models	RG	CS			CO	BLEU
	P%	P%	R%	F1%	DLD%	
CC (Wiseman et al., 2017)	74.80	29.49	36.18	32.49	15.42	14.19
NCP+CC (Puduppully et al., 2019a)	87.47	34.18	51.22	41.00	18.58	16.50
HierEnc (Gong et al., 2019a)	91.46	36.09	48.01	41.21	20.86	16.85
EdiNLG (Puduppully et al., 2019c)	91.41	30.91	**64.13**	41.71	21.72	17.01
MS-GPT-50 (Miculicich et al., 2019)	94.35	33.91	53.82	41.61	19.30	15.17
MS-End-to-End (Miculicich et al., 2019)	93.38	32.40	58.02	41.58	18.54	15.03
Systran-AI-Detok (Gong et al., 2019b)	84.16	34.88	43.29	38.63	22.72	**18.32**
NLE* (Saleh et al., 2019)	94.08	41.13	54.20	46.77	25.64	20.52
Hierarchical D2T (Rebuffel et al., 2020)	89.46	39.47	51.64	44.74	18.90	17.50
Stepwise HiBERT realized	95.88	41.49	53.86	46.87	18.10	14.79
Stepwise HiBERT planning only*	–	42.96	55.81	48.55	–	–
Stepwise ETCSum realized	**98.87**	**45.79**	58.49	**49.76**	**25.08**	17.56
Stepwise ETCSum planning only*	–	46.02	58.45	51.50	–	–

Table 2: Standard metrics for Rotowire: relation generation (RG) precision (P%), content selection (CS) precision (P%) and recall (R%), content ordering (CO) via the complement of normalized Damerau-Levenshtein distance (DLD%), and BLEU score. Models marked with a * are not directly comparable. Boldfaced numbers are the best results among comparable models.

extractive summarization task with a smaller model; BERTSum Large (Liu and Lapata, 2019b) with 340m parameters achieved 43.85/20.34/39.90 R1/R2/RL scores, whereas ours with 165m parameters achieved 43.84/20.80/39.77. Comparatively, Stepwise HiBERT did not do equally well on document summarization due to the sequential nature of the input. However, we demonstrate in Section 7 that it is well suited as an extractive content planner for table-to-text generation.

ROUGE scores in Table 1 are computed with a confidence interval of 95%. As such, Stepwise ETCSum(+TriBlk) is significantly better than BERTSum(+TriBlk), all variants of HierBERT, ETCSum and Stepwise RoBERTaSum(+TriBlk). For other models, such as RoBERTaSum(+TriBlk) and ETCSum+TriBlk, this confidence interval is not a deciding factor, hence we performed One-way ANOVA with posthoc Tukey-HSD tests ($p < 0.01$). Our best model Stepwise ETCSum performs significantly better than RoBERTaSum(+TriBlk), ETCSum+TriBlk and Stepwise ETCSum+TriBlk, on the average of ROUGE scores.

7 Table-to-Text Generation

Task. We further explore our model's ability to learn content plans for the Rotowire data-to-text generation task (Wiseman et al., 2017).[6] The task is to generate a summary of an NBA game from its box score (a table of statistics detailing the performance of the two teams and of each player). The dataset consists of 4853 pairs of box scores and summaries of NBA games played from 2014 to 2017. The data is split into 3398 train, 727 validation and 728 test examples. On average there are 628 records in a box score per game. The average summary has 337.1 words and 13.49 sentences.

Similar to Puduppully et al. (2019a) we decompose the problem into two sub-problems, which we solve independently: content planning, which consists of selecting which records in the table should be mentioned in the summary, in what order, and how they should be organized into sentences; and realization, which uses the content plan to create a human-readable summary. We refer the reader to the supplementary material for an example. Our main focus in this paper is to demonstrate our models' ability to model long and structured Rotowire input tables, and generate long meaningful content plans. For realization, we simply use a RoBERTa (Liu et al., 2019b) initialized sequence-to-sequence transformer model (Rothe et al., 2020), trained to emit the realization sentence by sentence.

We train our stepwise models to take a score table and the partially generated content plan, and predict the next element in the content plan. This can be either one of the entries in the score table, a sentence break or a token marking the end of the plan. Unlike extractive summarization, here an optimal extractive content plan can have repeated entries from the input table (e.g. team names) to better preserve and generate discourse relations among sentences in the target summary (Puduppully et al., 2019b), making it a challenging task for other iterative models that prohibit redundancy, e.g., (Bi et al., 2020). For details about model implementation, realization, and the induction of oracle content plans for training, we refer the reader to the supplementary material.

We report typical Rotowire metrics (Wiseman et al., 2017), using the standard information extraction system described by Puduppully et al. (2019a)

[6]The Rotowire dataset is available for download at https://github.com/harvardnlp/boxscore-data.

to extract the box score table relations mentioned in the generated (G) and in the target (T) summary. The metrics measure: text quality (BLEU score between G and T); relation generation quality (the precision of the relations extracted from G against the box score table); content selection quality (the precision and recall of the relations extracted from G against those extracted from T); and content ordering quality (the complement of the normalized Damerau-Levenshtein distance on the sequences of relations extracted from G and T). We also conducted human evaluation of Rotowire summaries.

Results. We focus on evaluating our Stepwise HiBERT and ETCSum models.[7] Our results are presented in Table 2. The "realized" scores assess the quality of our realized summaries and are comparable to systems in the first block in Table 2. We found both Stepwise HiBERT and Stepwise ETCSum do content selection particularly well. Their very high precision scores (41.49% and 45.79%, respectively) combined with good recall (53.86% and 58.49%, respectively) outperform Puduppully et al. (2019a) and other recent models on F1 score. In terms of content ordering and BLEU score, Stepwise HiBERT (14.79 BLEU, 18.10% DLD) performs worse than Puduppully et al. (2019a) (16.50 BLEU, 18.58% DLD), while Stepwise ETCSum performs significantly better (17.56 BLEU, 25.08% DLD). It's possible that a higher BLEU score could be achieved by improving our simple sentence-by-sentence realization method.

We also report content selection scores for the output of the content planning modules (see "planning only" models in Table 2). We drop name, city and date entries from our content plans before computing the metrics in order to make them comparable with others in Table 2. We see the roundtrip of realization and subsequent information extraction decreases CS quality slightly for both models (the absolute drop of F1 score is 1.68% for Stepwise HiBERT, and 1.74% for Stepwise ETCSum).

Human Evaluation. Participants were shown two summaries of an NBA game and asked to compare them with respect to *informativeness* (Does a summary present a better selection of the relevant facts about the game?) and *readability* (Which summary has a better narrative flow and is easier to read?). We randomly selected 50 NBA tables and evaluated summaries from Baseline (Wiseman et al., 2017), Stepwise HiBERT, Stepwise ETC and Gold. The average(max;min) number of sentences were 8(8;8), 12.7(17;9), 16.7(25;10) and 12.0(20;6), for Baseline, Stepwise HiBERT, Stepwise ETC, and Gold, respectively. We also included truncated summaries from Stepwise HiBERT and Stepwise ETC to match the number of sentences in corresponding Gold summaries. We elicited judgements from three different annotators for each pair. We report the Best(1)-Worst(-1) Scaling scores (Louviere and Woodworth, 1991; Louviere et al., 2015). Results are presented in Table 3.

Models	Informativeness	Readability
Baseline	0.06	**0.22**
Stepwise HiBERT	0.17	0.00
+truncated	-0.34	0.04
Stepwise ETCSum	**0.29**	-0.13
+truncated	-0.10	-0.08
Gold	-0.09	-0.03

Table 3: Human evaluation of Rotowire Summaries.

Overall, Stepwise ETC summaries were ranked most informative, but they performed worst on readability. The off-the-shelf sentence-level realizer (see the supplementary material) favors the statistics-dense sentences of the baseline summaries, as it tends to hallucinate on less dense plans. Future work will aim to address this limitation. For informativeness, Stepwise ETC summaries are significantly better than Gold, Stepwise ETC truncated and Stepwise HiBERT truncated summaries. Stepwise HiBERT summaries are significantly better than both truncated variants. All other differences are not significant ($p < 0.05$). For readability, baseline summaries are significantly better than both ETC variants and Stepwise HiBERT. All other differences are not significant.

8 Conclusion

The stepwise structured transformer paradigm, exemplified by HiBERT and ETCSum, can be easily adapted both to extractive document summarization or content planning for table-to-text generation. Stepwise ETCSum, in particular, sets a new standard for both tasks. Future work will focus on extending our models to generate extractive plans for better abstractive summarization of long or multiple documents (Liu et al., 2018).

[7] We don't reproduce BERTSum or RoBERTaSum baselines here for two reasons: i) these sequential models are not optimal for tabular data, and ii) they are also bounded by an input length of 512 tokens, the average length of linearized score tables is 7184 tokens per game. We also don't report on our non-stepwise models as they are not suitable to generate ordered content plans as required for this task.

Acknowledgments

We thank Joshua Ainslie and Santiago Ontanon for sharing their ETC code and checkpoints, and also giving us feedback on an early draft of this paper. We thank Annie Louis, the London and Zurich Generation teams, the reviewers and the action editor for invaluable feedback. We thank Enrique Alfonseca and Hadar Shemtov for their support for Longform Summarization.

References

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Reddy Ravula, and Sumit Sanghai. 2020. ETC: Encoding long and structured data in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (abs/2004.08483)*, Online.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Keping Bi, Rahul Jha, W. Bruce Croft, and Asli Celikyilmaz. 2020. AREDSUM: Adaptive redundancy-aware iterative sentence ranking for extractive document summarization. *CoRR*, abs/2004.06176.

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, New York, NY, USA.

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*, pages 484–494, Berlin, Germany.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, Georgia.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

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*, pages 4171–4186, Minneapolis, Minnesota.

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, Brussels, Belgium.

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.

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.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019a. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3143–3152, Hong Kong, China.

Li Gong, Josep Crego, and Jean Senellart. 2019b. SYSTRAN @ WNGT 2019: DGT Task. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 262–267, Hong Kong.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1315–1325, Minneapolis, Minnesota.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Karl Moritz Hermann, Tomáš Kočiský, 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 28*, 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. *CoRR*, abs/1805.06266.

Thomas Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *CoRR*, abs/2001.04451.

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

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, Portland, Oregon, USA.

Jingyun Liu, Jackie Chi Kit Cheung, and Annie Louis. 2019a. What comes next? Extractive summarization by next-sentence prediction. *CoRR*, abs/1901.03859.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver Canada.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy.

Yang Liu and Mirella Lapata. 2019b. 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*, pages 3730–3740, Hong Kong, China.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. *University of Alberta*, Working Paper.

Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. 2019. Reading like HER: Human reading inspired extractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3033–3043, Hong Kong, China.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on IR Research*, ECIR07, page 557564, Berlin, Heidelberg. Springer-Verlag.

Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins, and Shay B. Cohen. 2019. Jointly extracting and compressing documents with summary state representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3955–3966, Minneapolis, Minnesota.

Lesly Miculicich, Marc Marone, and Hany Hassan. 2019. Selecting, planning, and rewriting: A modular approach for data-to-document generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 289–296, Hong Kong.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016a. Classify or select: Neural architectures for extractive document summarization. *CoRR*, abs/1611.04244.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B. Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*, pages 2020–2030, Melbourne, Australia.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of*

the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 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.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy.

Ratish Puduppully, Jonathan Mallinson, and Mirella Lapata. 2019c. University of Edinburgh's submission to the document-level generation and translation shared task. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 268–272, Hong Kong.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. *CoRR*, abs/1911.05507.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *Advances in Information Retrieval*, pages 65–80, Cham. Springer International Publishing.

Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics, (abs/1907.12461)*, 8:264–280.

Aurko Roy, Mohammad Taghi Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *CoRR*, abs/2003.05997.

Fahimeh Saleh, Alexandre Berard, Ioan Calapodescu, and Laurent Besacier. 2019. Naver labs Europe's systems for the document-level generation and translation task at WNGT 2019. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 273–279, Hong Kong.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 464–468, New Orleans, Louisiana.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark.

Liqiang Xiao, Lu Wang, Hao He, and Yaohui Jin. 2020. Copy or Rewrite: Hybrid summarization with hierarchical reinforcement learning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3292–3303, Hong Kong, China.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 452–462, Vancouver, Canada.

Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. 2019. Bp-transformer: Modelling long-range context via binary partitioning. *CoRR*, abs/1911.04070.

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.

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.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019a. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1049–1058, Florence, Italy.

Ming Zhong, Danqing Wang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2019b. A closer look at data bias in neural extractive summarization models. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 80–89, Hong Kong, China.

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*, pages 654–663, Melbourne, Australia.

A Implementation and Reproducibility details

A.1 HiBERT

We did a wide range of hyperparameter search for HiBERT. We experimented with the number of layers in the document encoder ($1 < L_{\text{doc}} < 12$); the number of layers in the sentence encoder ($1 < L_{\text{sent}} < 12, L_{\text{doc}} < L_{\text{sent}}$); the initialization and sharing of position embeddings, $\mathbf{p}_j^{\text{token}}$, $\mathbf{p}_j^{\text{doc}}$ and $\mathbf{p}_j^{\text{sum}}$; the initialization and sharing of document and sentence encoder parameters with BERT and RoBERTa checkpoints; and the representation of sentence ("first token embedding" or "average of all token embeddings") from the sentence encoder.

For extractive summarization, we used HiBERT with a 8 transformer layer sentence encoder, and a 4 transformer layer document encoder. The model has 133,784,833 parameters. The word position embedding in the sentence encoder is initialized using the RoBERTa checkpoint, but the document and summary sentence position embeddings are learned from scratch. The document self attention and summary self attentions are shared and initialized using the RoBERTa checkpoint, the document-summary attention is also initialized using the RoBERTa checkpoint. We truncate each document to 128 sentences and each sentence to 32 words. We trained all HiBERT models for 100k steps saving checkpoints every 1000 steps, with a batch size of 32. Following Liu and Lapata (2019b), we choose the best model based on the MLE loss on the whole validation set.

For Rotowire, we use HiBERT with a 2 transformer layer sentence encoder, and a 4 transformer layer document encoder. The model has 91,448,065 trainable parameters. We don't use the document sentence position embeddings for Rotowire as the input consists of a set of entries in a table. We use the summary sentence position embedding to capture the order in the content plan. We use the ROBERTA vocabulary, but as discussed in B.3 we don't use ROBERTA pretraining, instead initializing with random weights. We trained the model with a batch size of 128 until the AUC score for predicting the next content plan entry on the validation dataset flattened out, which came after 766K steps. Since the dataset has 246290 examples (one for each element in the target content plan for each Rotowire example), the model saw the entire dataset approximately 398 times.

For all HiBERT models, we used Cloud TPU v3 accelerators for training and the Adam optimizer with a learning rate of 0.01.

A.2 ETCSum

The ETCSum model for both extractive summarization and table-to-text generation uses a 12 layer transformer as described in (Ainslie et al., 2020). The model is pretrained with MLM and CPC objectives as described in (Ainslie et al., 2020). In total, the model has 165,825,793 trainable parameters which mostly comes from the long input of 8192 tokens and the full attention of 512 of the global tokens. We trained our model with a batch size of 512 for 5,000 steps approximately equivalent to 10 epochs.

We used Cloud TPU v3 accelerators for training and inference was done on a V100GPU taking 10 hours to get predictions for the test set.

Model selection was done over models Rouge-1 performance in the validation set for all models except stepwise models where a subset of the validation set was used instead, consisting of the first 1000 examples, given the longer inference times.

We did a wide range of hyperparameter search where we experimented with learning rate (0.000025, 0.00005, 0.0001), relative position encoding vocabulary size (12, 24), the representation of sentences ("first token embedding" or "average

of all token embeddings") from the sentence encoder and in additionally non-stepwise models we experimented with positive label weight used to for loss calculation. Finally, we used an Adam optimizer with learning rate of 0.000025.

A.3 Realization model

We use a ROBERTASHARE model following (Rothe et al., 2020). The model has 152,491,008 trainable parameters. We trained the model until we reached the maximum BLEU score on validation data. We trained our model with a batch size of 512 for 36K steps. Since the dataset has 45533 examples (one for each element in the target content plan in each Rotowire example), the model saw the entire dataset approximately 405 times. We used Cloud TPU v3 accelerators for training. We used the Adam optimizer with a learning rate of 0.05.

B Table-to-Text Generation

B.1 Task

Table 4 shows a prototypical input table from the Rotowire dataset[8], along with a possible content plan and its realization. As shown in the example, a well-formed content plan can repeat some of the entries from the input table.

B.2 Generating Oracle Content Plans

The Rotowire dataset does not contain ground truth content plans for its summaries. Instead, we infer them following a similar approach to (Puduppully et al., 2019a), but with a few minor modifications: 1) we use just a single convolutional model, instead of an ensemble of convolutional models and LSTMs, 2) our plans maintain the within-sentence order of information, and may include repetitions if a piece of information is repeated within a sentence in the target summary, 3) our plans include sentence breaks, though we remove sentences with no table entries, 4) our content plans can include the match date, if it's mentioned in the text (e.g. "on Saturday"), 5) when we resolve a pronoun, we emit the corresponding player or team name to the content plan. With respect to Table 4, if the realization at the bottom was a reference summary, then by applying this process we would obtain the content plan shown in the middle of the table. On average, the plans inferred in this fashion have 59.24 table entries and 12.72 sentences.

[8]We are not presenting an actual example for legal reasons.

B.3 Content planning technical details

HiBERT. Conceptually, the input to HiBERT is a sequence of strings. We use three special strings, i.e., `<BEG>`, `<EOS>`, `<EOT>`, to explicitly mark the beginning of the content plan, the end of a sentence, and the end of the plan (text), respectively. The other strings are the values from the table, e.g., `Chicago_Bulls Points`, in the same order in which they appear in the text. In practice, in an attempt to leverage ROBERTA pre-training, we replace value strings with natural language sentences that we generate from each value using the templates listed in Table 5. For numeric values, such as the number of points of a team or player, similarly to Puduppully et al. (2019c) we compute the rank of the value among the instances of the same table entry type, and include that in the templated sentence in the form of a "which is [1st, 2nd, 3rd, ..., Nth] best" suffix[9]. With respect to the example in Table 4, the value `Chicago_Bulls Points` would then be represented as the natural language sentence: "team points scored of Chicago_Bulls is 100 which is 1st best". As we did not observe a significant benefit in terms of AUC when predicting the next content plan entry on validation data, we eventually initialized our model with random weights but retained the natural language representation of the value strings.

Because HiBERT has a sentence limit of 512, we do a pre-filtering step by discarding the table entries that are less likely to be mentioned in the summary, i.e., all player entries valued "N/A" and as many entries valued "0" as needed. Since the table entries aren't naturally ordered we don't feed a positional embedding $\mathbf{p}_i^{\text{sent}}$ in the document encoder, but we still feed it for the summary encoder.

Given the table entries and partial summary, HiBERT computes a distribution over the input sentences, where `<EOS>` corresponds to emitting a sentence break, `<EOT>` corresponds to ending the content plan, and `<BEG>` is not used.

We sample content plans from a trained model by greedy decoding with one modification: entries are not allowed to repeat in the content plan, except for sentence breaks, team names and team cities. If the highest probability sentence would have been a repeat, we instead emit the second highest, etc.

[9]We use the words "which is Nth best" even when a high number is logically detrimental to the team (e.g. when it represents losses).

Input Table: Match date: Saturday, 22nd October 2018								
Team	Name	City	At home?	Wins	Losses	Points	Rebounds	...
Chicago_Bulls	Bulls	Chicago	Home	3	1	100	21	...
LA_Lakers	Lakers	Los_Angeles	Away	2	5	80	25	...
Player	Name	Surname	Team	Points	Rebounds	Assists	...	
Michael_Jordan	Michael	Jordan	Chicago_Bulls	25	10	10	...	
Shaquille_O_Neal	Shaquille	O'Neal	LA_Lakers	30	15	11	...	
...	...	...	...	...	...	...	...	

Content Plan

S1: Chicago_Bulls city, Chicago_Bulls name, LA_Lakers city, LA_Lakers name, Chicago_Bulls points, LA_Lakers points, Match date, EOS.
S2: LA_Lakers name, LA_Lakers wins, LA_Lakers losses, Shaquille_O_Neal surname, Shaquille_O_Neal points, EOS.
S3: Chicago_Bulls city, Chicago_Bulls name, Chicago_Bulls wins, Chicago_Bulls losses, Michael_Jordan name, Michael_Jordan surname, Michael_Jordan points, Michael_Jordan rebounds, EOS.

Realization

S1: The Chicago Bulls won against the Los Angeles Lakers 100 - 80 on Saturday.
S2: It was a poor showing for the Lakers (2 - 5) in spite of O'Neal's 30 point contribution.
S3: The Chicago Bulls' (3 - 1) best player was, predictably, Michael Jordan with 25 points and 10 rebounds.

Table 4: An hypothetical example from the Rotowire dataset for an NBA game, possible 3-sentence content plan and corresponding 3 realized sentences below.

ETCSum. ETC models used the same filtered set of table entries used in HiBERT as input. We concatenated these entries into a flat input sequence. Similarly, we used special strings <EOS>, <EOT> and <BEG> which correspond to the same concepts as in HiBERT, end of sentence, end of text and beginning of text respectively. These special strings are appended at the beginning of the flat input sequence.

The partial summary input is constructed by concatenating the special string <BEG> and the entries that have been predicted so far, in order of prediction, with <EOS> indicating sentences breaks.

The full input sequence is then constructed by concatenating: a [CLS] delimiter, the flat input sequence, a special separator [SEP], the partial summary and finally a separator [SEP]. Both the input sequence and the partial summary are padded to 6141 and 2048 respectively, adding up in total to 8192 strings for the full input, including the special delimiters.

The model uses additional inputs to construct the global-local attention. One global token is assigned to each segment in the full input, each special delimiter gets assigned a global token, as well as every sentence in the input and partial summary. The model has a maximum global token id of 512, this has to be taken into account for examples where the number of segments, input sequence sentences, special delimiters and partial inputs is larger than 512. For those examples, we don't assign global tokens to the tail of the input sequence.

To be consistent we use the same decoding strategy where we sample content plans greedily but without repeated entries allowed in the content plan except for sentence breaks, team names and team cities.

B.4 Rotowire realization model

The generated content plans are realized via a sequence-to-sequence transformer model initialized with ROBERTA (Liu et al., 2019b) following (Rothe et al., 2020), trained to emit the realization sentence by sentence. The input to the model is the concatenation of the following:

1. The text of the previous sentence, or the empty string (for the first sentence). (The model can use this to pronominalize team and player names if they were already introduced.)
2. The literal string " <BEG> " as a separator.
3. The templated realizations (cfr. Table 5) of the entries in the sentence's content plan, space separated.
4. The literal string " <CONTEXT> " as a separator.
5. The templated representation of the match date.
6. For both teams, the templated representations of a) the team name, b) the team city, c) TEAM-PTS, d) TEAM-WINS, e) TEAM-LOSSES, f) whether the team was playing at home or away. These are space separated.
7. For each player **in the sentence's content plan**: the templated representations of a) PLAYER-START_POSITION, and b) which team the player was on. These are space separated.

The input after the " <CONTEXT> " separator is provided because we noticed that sometimes the content plan doesn't provide all the necessary information for realizing a sentence. For example, sometimes the target text may refer to a player

Table entry type	Template used
match date	match date of match is year: YYYY month: MM day: DD day_of_week: W
team name	team name of T is V
team city	team city of T is V
TEAM-PTS_QTR1	team 1st quarter points of T is V
TEAM-PTS_QTR2	team 2nd quarter points of T is V
TEAM-PTS_QTR3	team 3rd quarter points of T is V
TEAM-PTS_QTR4	team 4th quarter points of T is V
TEAM-FT_PCT	team free throw percentage of T is V
TEAM-PTS	team points scored of T is V
TEAM-AST	team assists of T is V
TEAM-LOSSES	team losses of T is V
TEAM-WINS	team wins of T is V
TEAM-REB	team rebounds of T is V
TEAM-TOV	team turnovers of T is V
TEAM-FG3_PCT	team 3-point field goal percentage of T is V
TEAM-FG_PCT	team field goal percentage of T is V
team playing at home or away?	T is home/away team of match
player first name	player first name of P is V
player second name	player second name of P is V
PLAYER-PTS	player points scored of P is V
PLAYER-FGM	player field goals made of P is V
PLAYER-FGA	player field goals attempted of P is V
PLAYER-MIN	player minutes played of P is V
PLAYER-FG3M	player 3-point field goals made of P is V
PLAYER-FG3A	player 3-point field goals attempted of P is V
PLAYER-STL	player steals of P is V
PLAYER-FTM	player free throws made of P is V
PLAYER-FTA	player free throws attempted of P is V
PLAYER-BLK	player blocks of P is V
PLAYER-AST	player assists of P is V
PLAYER-TO	player turnovers of P is V
PLAYER-PF	player fouls of P is V
PLAYER-REB	player rebounds of P is V
PLAYER-START_POSITION	player starting position of P is V
PLAYER-OREB	player offensive rebounds of P is V
PLAYER-DREB	player defensive rebounds of P is V
PLAYER-FG_PCT	player field goals percentage of P is V
PLAYER-FG3_PCT	player 3-point field goals percentage of P is V
PLAYER-FT_PCT	player free throws percentage of P is V
the team a player belongs to	P is player of T

Table 5: The templates we use to create textual representations of the table entries. In the templates, YYYY, MMM, DD, W, T, P and V are placeholders. W encodes the day of week: Monday is 0, Sunday is 6. X is the name of a team or of a player. V is the value that the team (T) or player (P) has for the given table entry in the dataset. The names in the table entry column correspond to the names of properties in the Rotowire dataset where possible.

Models	RG	CS			CO	BLEU
	P%	P%	R%	F1%	DLD%	
Stepwise HiBERT realized	95.97	41.34	57.62	48.14	19.19	15.86
Stepwise HiBERT planning only*	–	42.83	59.62	49.85	–	–
Stepwise ETCSum realized	98.78	45.18	60.14	51.60	25.87	17.93
Stepwise ETCSum planning only*	–	45.53	60.14	51.82	–	–

Table 6: Standard metrics for Rotowire on validation data.

by their starting position and team, which is information that wouldn't otherwise be provided to the realizer.

We create training data from the rotowire summaries and their inferred content plans by splitting them into sentences together with our inferred content plans. We realize content plans by autoregressively feeding the sentence produced in the previous step as input to the next step.

B.5 Validation data performance

We report performance of our best models on the Rotowire validation data in Table 6.