Generating Multi-Sentence Abstractive Summaries of Interleaved Texts

Sanjeev Kumar Karn[1,3], Francine Chen[2], Yan-Ying Chen[2], Ulli Waltinger[3] and Hinrich Schütze[1]

[1]Center for Information and Language Processing (CIS), LMU Munich
[2]FX Palo Alto Laboratory, Palo Alto, California
[3]Machine Intelligence, Siemens CT, Munich, Germany
[1]skarn@cis.lmu.de
[2]{chen,yanying}@fxpal.com

Abstract

In multi-participant postings, as in online chat conversations, several conversations or topic threads may take place concurrently. This leads to difficulties for readers reviewing the postings in not only following discussions but also in quickly identifying their essence. A two-step process, disentanglement of interleaved posts followed by summarization of each thread, addresses the issue, but disentanglement errors are propagated to the summarization step, degrading the overall performance. To address this, we propose an end-to-end trainable encoder-decoder network for summarizing interleaved posts. The interleaved posts are encoded hierarchically, i.e., word-to-word (words in a post) followed by post-to-post (posts in a channel). The decoder also generates summaries hierarchically, thread-to-thread (generate thread representations) followed by word-to-word (i.e., generate summary words). Additionally, we propose a hierarchical attention mechanism for interleaved text. Overall, our end-to-end trainable hierarchical framework enhances performance over a sequence to sequence framework by 8% on a synthetic interleaved texts dataset.

1 Introduction

Interleaved texts are becoming more common with new ways of working and new forms of communication, e.g., multi-author entries for activity reports, meeting minutes and social media conversations, such as Slack. Quickly getting a sense of or following the content of different threads in interleaved texts, where posts belonging to different threads occur in one sequence, is often difficult. An example of two threads with multiple posts interleaved to form a sequence is:

Post1-Thread1 → Post1-Thread2 →
Post2-Thread1 → Post3-Thread1 →
Post2-Thread2 → Post3-Thread2

This intermingling leads to difficulties in not only following discussions but also in retrieving their essence. In conversation disentanglement, interleaved posts are grouped by the thread; e.g., the previous example could be rearranged as:

Post1-Thread1 → Post2-Thread1 →
Post3-Thread1
Post1-Thread2 → Post2-Thread2 →
Post3-Thread2

In analyzing interleaved texts, Shang et al. (2018) went a step further and proposed summarization of the interleaved texts. They designed an unsupervised two-step system and evaluated the system on meeting texts. In the first step, a conversation disentanglement component disentangles the texts thread-wise. Then, in the second step, a multi-sentence compression component compresses the thread-wise posts to single sentence summaries. However, this system has a major disadvantage, in that the disentanglement obtained through either supervised (Jiang et al., 2018) or unsupervised (Wang and Oard, 2009) methods propagate its errors to the downstream summarization task, and thus, degrades the overall performance.

We aim to tackle this issue of error propagation through an end-to-end trainable encoder-decoder system that takes a variable length input, e.g., interleaved texts, processes it and generates a variable length output, e.g., a multi-sentence summary; see Figure 1 for an illustration. An end-to-end system eliminates the disentanglement component, and thus, the error propagation.

The encoder first performs word-to-word encoding to embed each post, followed by post-to-post encoding to embed the overall content of the posts and represent the discourse structure of the interleaved texts. The decoder has a thread-to-thread decoder to generate a representation for

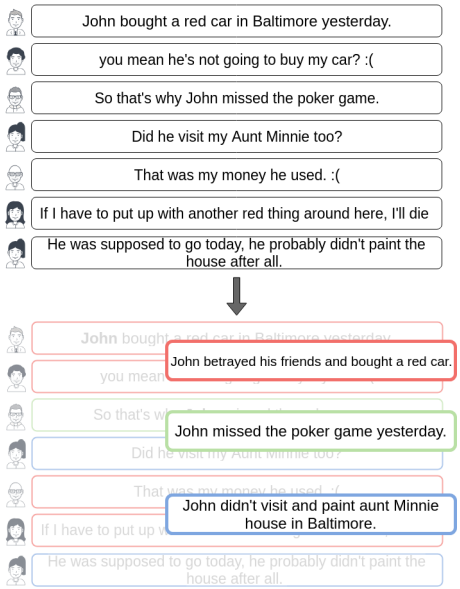

Figure 1: 7 interleaved posts are implicitly disentangled into 3 threads, and single sentence summaries are generated for each thread. Posts are outlined with colors corresponding the color of each summary.

each thread, and the thread representation is given to a word-to-word decoder to generate a summary sentence. We also propose to use hierarchical attention similar to Nallapati et al. (2017), but instead of computing sentence-level attention at every word, attentions are only computed when decoding new sentences. Further, the attention networks are trained end-to-end.

Despite the availability of a multitude of real-world interleaved texts, a major challenge to train encoder-decoder models is the lack of labels (summaries). As labeling is difficult and expensive (Barker et al., 2016; Aker et al., 2016; Verberne et al., 2018), we synthesize a corpus by mixing texts and associated summaries from a corpus of documents, where the mixed text has a structure reflective of a multi-party conversation with interleaved threads and the summary highlights the information in the threads. We find abstracts and titles of randomized controlled trial (RCT) articles, a PubMed corpus, fit for the purpose.

To summarize, our contributions are threefold:

- We propose to combine a hierarchical encoder and decoder to obtain multi-sentence summaries of interleaved texts.
- We propose a novel hierarchical attention mechanism that is integrated with the hierarchical encoder-decoder architecture and which equips the decoder to disentangle the interleaving further.
- We use a synthetic dataset to verify the ideas and show our end-to-end trainable architecture addresses not only the issue of error-propagation observed in competitive methods but also improves the performance.

2 Related Work

Quite often multi-party conversations, e.g. news comments, social media conversation and activity report, have tens of posts discussing several different matters pertaining to a subject. Ma et al. (2012); Aker et al. (2016); Shang et al. (2018) designed methodologies to summarize posts in order to provide an overview on the discussed matters. They broadly follow the same approach: cluster the posts and then extract a summary from each cluster.

There are two kinds of summarization: abstractive and extractive. In abstractive summarization, the model utilizes a corpus level vocabulary and generates novel sentences as the summary, while extractive models extract or rearrange the source words as the summary. Abstractive models based on neural sequence-to-sequence (seq2seq) (Rush et al., 2015) proved to generate summaries with higher ROUGE scores than the feature-based abstractive models. Integration of attention into seq2seq (Bahdanau et al., 2014) led to further advancement of abstractive summarization (Nallapati et al., 2016; Chopra et al., 2016).

There are many possible patterns of organization of the information in texts, e.g., chronological pattern. News articles have an inverted pyramid pattern, i.e., core information in the lead sentences and the extra information in later sentences. A seq2seq model is appropriate for summarization of a news article as it encodes and decodes sequentially. However, in interleaved texts, related information maybe separated; thus a seq2seq model may be competent but not sufficient.

Li et al. (2015) proposed an encoder-decoder (auto-encoder) model that utilizes a hierarchy of networks: word-to-word followed by sentence-to-sentence. Their model is better at capturing the underlying structure than a vanilla sequential encoder-decoder model (seq2seq). Krause et al. (2017); Jing et al. (2018) showed multi-sentence captioning of an image through a decoder based on a hierarchical Recurrent Neural Network (RNN),

topic-to-topic followed by word-to-word, is better than seq2seq.

These works suggest a hierarchical encoder, with word-to-word encoding followed by post-to-post, will better recognize the dispersed information in interleaved texts. Similarly, a hierarchical decoder, thread-to-thread followed by word-to-word, will intrinsically disentangle the posts, and therefore, generate more appropriate summaries.

Nallapati et al. (2016) devised a hierarchical attention mechanism for a seq2seq model, where two levels of attention distributions over the source, i.e., sentence and word, are computed at every step of the word decoding. Based on the sentence attentions, the word attentions are rescaled. Hsu et al. (2018) slightly simplified this mechanism and computed the sentence attention only at the first step. Our hierarchical attention is more intuitive and computes new sentence attentions for every new summary sentence, and unlike Hsu et al. (2018), is trained end-to-end .

3 Model

Problem Statement

We aim to design a system that when given a sequence of posts, $C = \langle P_1, \ldots, P_{|C|} \rangle$, produces a sequence of summaries, $T = \langle S_1, \ldots, S_{|T|} \rangle$. For simplicity and clarity, unless otherwise noted, we will use lowercase italics for variables, uppercase italics for sequences, lowercase bold for vectors and uppercase bold for matrices.

Figure 2 illustrates the proposed hierarchical encoder-decoder framework. In the framework, first, a low-level, word-to-word encoder converts a sequence of words in a post, P_j, to a sequence of representations, $H_j = \langle \mathbf{h}_{j0}, \ldots, \mathbf{h}_{j|P_j|} \rangle$. Subsequently, a top-level, post-to-post encoder converts those representations, $\langle H_0, \ldots, H_{|C|} \rangle$, to a sequence of top-level post representations $\langle \mathbf{m}_1, \ldots, \mathbf{m}_{|C|} \rangle$. These encoded representations are then passed to the decoder, which utilizes a top-level, thread-to-thread, decoder to disentangle them into a sequence of thread representations $\langle \mathbf{s}_1, \ldots, \mathbf{s}_{|T|} \rangle$. Finally, a low-level, word-to-word, decoder takes a thread representation $\mathbf{s}_i$ and generates a sequence of summary words $\langle w_{i1}, \ldots, w_{i|S_i|} \rangle$.

The maximum number of posts in a sequence of interleaved texts is denoted by n and threads by m. We limit the number of words in posts and summaries to fixed lengths by either truncating or padding, and denote them by p and q respectively. The hidden states of the encoder and decoder have dimensionality l.

3.1 Encoder

The hierarchical encoder used in Figure 2 is based on Nallapati et al. (2017), where word-to-word and post-to-post encoders are bi-directional LSTMs. We refer to Nallapati et al. (2017) for further details.

3.2 Decoder

Two major decoding approaches have been utilized in multi-sentence image captioning. Jing et al. (2018) and Krause et al. (2017) use a top-level RNN, such as thread-to-thread in our case, that takes an image representation and computes topic representations, and a low-level RNN, such as word-to-word in our case, that takes those topic representations and generates sentences corresponding to them. However, a major issue with their systems is the repetition of some of the sentences in a multi-sentence caption. To address this issue, Xue et al. (2018) proposed a top-level network instead of an RNN, that at each step not only takes an image but also an encoded representation of the previously generated caption sentence and then computes a topic representation. In summary, Jing et al. (2018) and Krause et al. (2017) use a top-level decoder which keeps track of its regions to generate a new topics, while Xue et al. (2018) uses a top-level decoder which takes feedback from the low-level decoder to generate a new topic.

We implement and experiment with both types of decodings, but apply it to text input. We refer to the former as threadLSTM and latter as feedbackLSTM. For details, refer to Jing et al. (2018); Krause et al. (2017); Xue et al. (2018). As our input is interleaved texts rather than images, there are a few fundamental changes in the decoders that will be described below.

3.2.1 threadLSTM

As discussed, the threadLSTM is hierarchical, in which the top level is a thread-to-thread decoder (D_{t2t}) and the low level is a word-to-word decoder (D_{w2w}) (see the right side in Figure 2).

A thread-to-thread decoder is a unidirectional LSTM ($f^{D_{t2t}}$) with its initial state $\mathbf{h}_0^{D_{t2t}}$ set with the feedforward-mapped conversation vector $\mathbf{c}'$. The input to the single layer feedforward is the

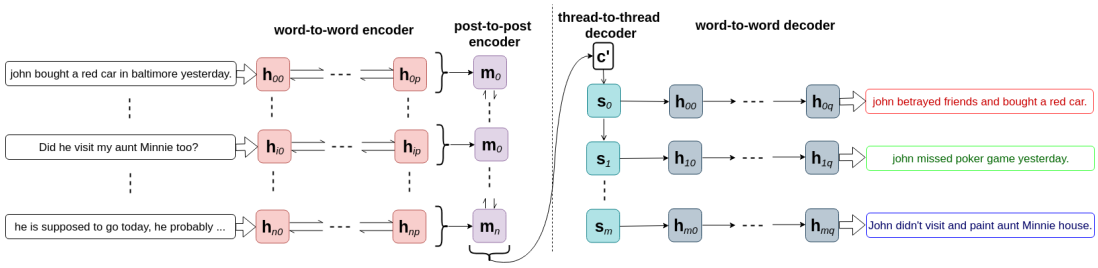

Figure 2: Hierarchical encoder-decoder architecture.

last state of the post-to-post encoder. At any step i of the decoder, a sequence of attention weights, $\langle \beta_{i0}, \ldots, \beta_{in} \rangle$, corresponding to the post representations, $\langle \mathbf{m}_0, \ldots, \mathbf{m}_n \rangle$, are computed utilizing the previous state, $\mathbf{h}_{i-1}^{D_{t2t}}$:

$$\mathbf{e}_{ij} = \text{top\_attn}(\mathbf{h}_{i-1}^{D_{t2t}}, \mathbf{m}_j) \quad (1)$$

$$\beta_{ij} = \sigma(\mathbf{e}_{ij}) \quad (2)$$

where top_attn is a single layer feedforward that aligns the previous state $\mathbf{h}_{i-1}^{D_{t2t}}$ to a post representation $\mathbf{m}_j$ and a sigmoid over the resulting value computes an attention weight β_{ij}. The left-hand side in Figure 3 depicts the process.

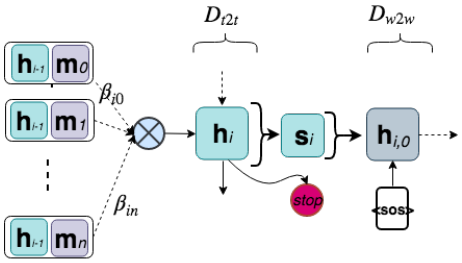

Figure 3: A step in the thread-to-thread decoder.

A weighted representation of the posts (crossed blue circle) is then computed: $\frac{1}{n}\sum_{j=1}^{n}\beta_{ij}\mathbf{m}_j$, and used as the next input to LSTM $f^{D_{t2t}}$, which then uses the previous state and this input to compute the next state $\mathbf{h}_i^{D_{t2t}}$.

The current state $\mathbf{h}_i^{D_{t2t}}$ is passed through a single layer feedforward network and a distribution over STOP=1 and CONTINUE=0 is computed:

$$p_i^{STOP} = \sigma(\text{g}\left(\mathbf{h}_i^{D_{t2t}}\right)) \quad (3)$$

where g is a feedforward network. In Figure 3, the process is depicted by a pink circle. The thread-to-thread decoder keeps decoding until p_i^{STOP} is greater than 0.5.

Additionally, the current state $\mathbf{h}_i^{D_{t2t}}$ is passed through another single layer feedforward network k followed by $\tanh$ activation to compute the threads representation $\mathbf{s}_i = \tanh(\text{k}\left(\mathbf{h}_i^{D_{t2t}}\right))$

Given a thread representation $\mathbf{s}_i$, the word-to-word decoder generates a summary for the thread. Our word-to-word decoder is based on Bahdanau et al. (2014). It is a unidirectional attentional LSTM ($f^{D_{w2w}}$); see the right-hand side of Figure 2. We refer to Bahdanau et al. (2014) for further details.

3.2.2 feedbackLSTM

The Xue et al. (2018) top-level decoder has a separate LSTM that runs over the words of the previously generated sentence in order to obtain a feedback signal for a newer topic; however, the same feedback signal could also be attained by utilizing the last hidden state of the low-level decoder that generated the previous sentence. This removes the overhead of running an extra LSTM, and therefore, speeds up the training. Thus, we use the last hidden state $\mathbf{h}_{i-1.}^{D_{w2w}}$ of unidirectional word-to-word ran on the previously generated summary to compute a thread representation, $\mathbf{s}_i$. Overall, converting a threadLSTM to a feedbackLSTM requires removal of the thread-to-thread decoder and replacing Eq. 1 with Eq. 4.

$$\mathbf{e}_{ij} = \text{top\_attn}(\text{u}(\mathbf{h}_{i-1q}^{D_{w2w}}), \mathbf{m}_j) \quad (4)$$

where, u is a single layer feed-forward network. Since there is no thread-to-thread decoder, the top-level attentions, $\boldsymbol{\beta}_.$, is computed using Eq. 4 and consequently the weighted average is passed through a feed-forward network to compute a thread representation $\mathbf{s}_i = \tanh(\text{k}\left(\frac{1}{n}\sum_{j=1}^{n}\beta_{ij}\mathbf{m}_j\right))$

The word-to-word decoder remains the same as the word-to-word decoder of Section 3.2.1. Decoding of summaries stops when the summary

contains only the dummy $\langle END \rangle$ (End Of Summary).

3.3 Hierarchical Attention

Our novel dynamic hierarchical attention is applicable to hierarchical decoding and is indifferent to encoding methodologies. The attention mechanism reutilizes the higher level attentions, $\boldsymbol{\beta}$, that are computed while obtaining a thread representation, $\mathbf{s}$, and scales the lower level attentions, $\boldsymbol{\alpha}$, that are computed while generating a word, y, of a summary, S; see Figure 4.

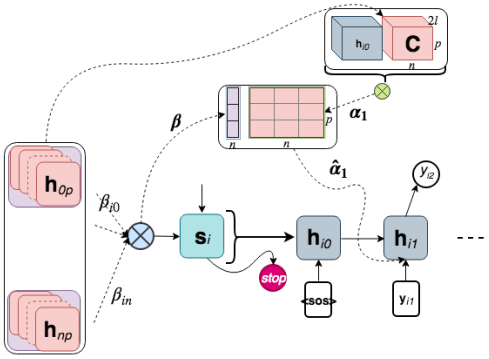

Figure 4: Hierarchical attention mechanism for a hierarchical encoder.

In case of a hierarchical encoder, attention weights, $\boldsymbol{\alpha}_{j\cdot}$, computed while decoding a word, y_j, utilize the post-level attentions, $\boldsymbol{\beta}$. Thus, the new word level attention weights, $\hat{\boldsymbol{\alpha}}_{j\cdot}$, are computed as below:

$$\hat{\alpha}_{jtk} = \frac{\beta_t \times \exp(\mathbf{e}_{jtk})}{\sum_{t=1}^{n} \beta_t \times (\sum_{k=1}^{p} \exp(\mathbf{e}_{jtk}))} \tag{5}$$

$$\mathbf{e}_{jtk} = \text{attn}(\mathbf{h}_{j-1}^{D_{w2w}}, \mathbf{C}_{tk\cdot}) \tag{6}$$

where $\mathbf{e}_{jtk}$ is computed as in Eq. 6. $\mathbf{C}$ in Eq. 6 is a word-to-word encoder representation matrix of dimension $n \times p \times 2l$, and attn is a feedforward network that aligns the current word decoder state $\mathbf{h}_{j-1}^{D_{w2w}}$ with all $n \cdot p$ representation vectors in $\mathbf{C}$.

In case of a sequential encoder, there is only one encoder, i.e., a word-to-word encoder, and the resulting word representation matrix, $\mathbf{C}$, has a dimension of $n \cdot p \times 2l$. In Eq. 1, the decoder state is aligned to a word representation, $\mathbf{w}_\cdot$, instead of a thread representation, and therefore, the resulting higher level attentions, $\boldsymbol{\beta}$, is of size $n \cdot p$. The hierarchical attention uses these high-level word attentions, $\boldsymbol{\beta}$, for rescaling a low level attention:

$$\hat{\alpha}_{jk} = \frac{\beta_k \times \exp(\mathbf{e}_{jk})}{\sum_{k=1}^{n \cdot p} \beta_k \times \exp(\mathbf{e}_{jk})} \tag{7}$$

3.4 Training Objective

We train our hierarchical encoder-decoder network similarly to an attentive seq2seq model (Bahdanau et al., 2014). Given a summary, $Y_i = \langle w_{i0}, \ldots, w_{iq} \rangle$, our word-to-word decoder generates a target $\hat{Y} = \langle y_{i0}, \ldots, y_{iq} \rangle$, with words from a same vocabulary U. We train our model end-to-end by minimizing the objective given in Eq. 8.

$$\frac{1}{(n \cdot q)} \sum_{i=1}^{n} \sum_{j=1}^{q} \log p_\theta \left(y_{ij} | w_{i \cdot < j}, \mathbf{C} \right) \tag{8}$$

4 Dataset

Obtaining labeled training data for conversation summarization is challenging. The available ones are either extractive (Verberne et al., 2018) or too small (Barker et al., 2016; Anguera et al., 2012) to train a neural model. To get around this issue and verify the proposed architecture, we synthesized a dataset by utilizing a corpus of conventional texts for which summaries are available. The PubMed corpus contains a type of article, i.e., randomized controlled trials (RCT), where sentences in the abstract are structured into sections and the title of the article summarizes the information from these sections (Dernoncourt and Lee, 2017). Thus, a random interleaving of sentences from a few abstracts roughly resembles interleaved texts, and correspondingly interleaving of titles resembles its multi-sentence summary. We devised an algorithm for creating the synthetic interleaved texts; see Algorithm. 1.

In Algorithm. 1, INTERLEAVE takes a set of concatenated abstracts and titles, $C = \langle A_1; T_1, \ldots, A_{|C|}; T_{|C|} \rangle$, minimum, a, and maximum, b, number of abstracts to interleave, and minimum, m, and maximum, n, number of sentences in an abstract, and then returns a set of concatenated interleaved texts and summaries. WINDOW takes a sequence of texts, X, and returns another sequence of texts, Y, of size $\frac{|X|-w}{t} + 1$, where w and t are window size and sliding step respectively. *window* reuses elements of X, and therefore, enlarges the corpus size. Notations $\mathcal{U}$ refers to a uniform sampling, $[\cdot]$ to array indexing, ADD to adding of elements and ; to concatenating.

Algorithm 1 Interleaving PubMed RCT abstracts

procedure INTERLEAVE(C, a, b, m, n)
  $\hat{C} \leftarrow$ WINDOW($C, w = b, t = 1$)
  $Z \leftarrow$ Array()
  for i = 1 to $|\hat{C}|$ **do**
    $T', A' \leftarrow$ Array(), Array()
    $S \leftarrow$[1, . . ., b]
    $r \sim \mathcal{U}(a, b)$
    for j = 1 to r **do** ▷ Selection
      $y \leftarrow \mathcal{U}(S)$
      $A, T \leftarrow \hat{C}[j]$
      T'.ADD(T)
      $q \sim \mathcal{U}(m, n)$
      A'.ADD(A[1:q])
      $S \leftarrow S \backslash \{y\}$
    $\hat{T}, \hat{I} \leftarrow$ Array(), Array()
    $S \leftarrow$[1, . . ., r]
    while True **do** ▷ Interleaving
      $k \leftarrow \mathcal{U}(S)$
      if $A'[k] == \emptyset$ **then**:
        if $\{j \in [1, \ldots, r] \mid A'[j] \neq \emptyset\} == \emptyset$ **then**:
          return False ▷ Sentences Exhausted
        else:
          Continue
      $I \leftarrow \hat{A}[k][1]$
      $A'[k] \leftarrow A'[k] \backslash \{I\}$
      $\hat{I}$.ADD(I)
      $T \leftarrow T'[k]$
      if $T \notin T'$ **then**:
        $\hat{T}$.ADD(T)
    Z.ADD($\hat{I}$;$\hat{T}$)
  return Z

We vary INTERLEAVE parameters as below, and create three different corpora for experiments:

- Easy: a=2, b=2, m=5 and n=5
- Medium: a=2, b=3, m=2 and n=5
- Hard: a=2, b=5, m=2 and n=5

Table 1 shows an example of a data instance in the Hard Corpus.

RCT articles have MeSH descriptors[1] that categorizes them into 16 categories. We use this information to split the above corpora into Train, Validation and Test. Training uses 14 categories and one each is used for test and validation. We didn't do any hyper-parameter tuning; however, we use validation for early stopping.

5 Experiments

Evaluation Metrics: we report ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (longest-common substring) as the quantitative evaluation of the models.

[1] https://meshb.nlm.nih.gov/treeView

A_2	media detective is a 10-lesson elementary school substance use prevention program developed on the basis of the message . . .
A_0	the primary purpose of this study was to conduct a randomized effectiveness trial of multisystemic therapy for child abuse and neglect. . .
A_2	the purpose of this study was to conduct a short-term , randomized , controlled trial to evaluate the effectiveness of media. . .
A_0	eighty-six families being followed by child protective services due to physical abuse were randomly assigned to mst-can. . .
A_2	elementary schools were randomly assigned to conditions to either receive the media detective program (n=344). . .
A_1	this study describes the development and testing of a multicomponent media campaign aimed at increasing discussions of alcohol. . .
	⋮
T_2	media literacy education for elementary school substance use prevention : study of media detective .
T_0	multisystemic therapy for child abuse and neglect : a randomized effectiveness trial .
T_1	description of a media campaign about alcohol use during pregnancy .

Table 1: The top rows contain interleaving of 3 articles with 2 to 5 sentences, bottom rows contain Multi-sentence Summary (3 interleaved titles).

Parameters: We initialized all weights, including word embeddings, with a random normal distribution with mean 0 and standard deviation 0.1. The embedding vectors are of dimension 100. The vocabulary size is limited to 8000. All hidden states of the encoder and decoder in the models are set to dimension 200. We pad short sequences with a special token, $\langle PAD \rangle$. We use Adam (Kingma and Ba, 2014) with an initial learning rate of .0001 and batch size of 64 for training. Texts are lowercased. In seq2seq experiments, the number of steps in the source is limited to 650 and the target to 75. For the word-to-word encoder, the steps are limited to 30, while the steps in the word-to-word decoder are limited to 15. The steps in the post-to-post encoder and thread-to-thread decoder depend on the corpus type, e.g., Medium has 15 steps in post-to-post and 3 steps in thread-to-thread. Train, test and validation have approximately 290k, 6k and 1.5k instances, respectively.

5.1 Baseline

We reimplemented the Bahdanau et al. (2014) seq2seq model, and evaluated it on a standard task of news summarization. We use a popular, CNN/DailyMail, dataset (Napoles et al., 2012), with ≈300k training examples for the purpose. We fetched the test set from See et al. (2017) and report the results on it. The results are compared with the seq2seq methods of Nallapati et al.

(Nallapati et al., 2016) and See et al. (See et al., 2017). As our aim for this experiment is to demonstrate the strength of the reimplementation, we set the parameters to produce comparable results in less computation time (2 days compared to others weeks). Table 2 compares performances, and the results indicate that our unenhanced implementation is competitive to the latter enhanced models.

Model	Rouge-1	Rouge-2	Rouge-L
seq2seq (Nallapati et al.)	32.49	11.84	29.47
seq2seq (See et. al.)	31.33	11.81	28.83
seq2seq (Ours)	29.58	09.36	20.21

Table 2: Rouge F1 on the standard task of Abstractive News Summarization (CNN/DailyMail). "Ours" is a reimplementation of Bahdanau et al. (2014).

We then take the implemented seq2seq model and train and test it on the Easy corpus. We also ran Shang et al. (2018)'s two-step system on the test set of the Easy corpus. As the Shang et al. (2018) system is unsupervised, it doesn't need training. Additionally, we also utilized Shang et al. (2018)'s clustering component to first cluster the interleaved texts of the corpus, and then the disentangled corpus is used to train the seq2seq model. We refer to the latter as cluster→seq2seq. The performance comparison of Shang et al. (2018) and the two seq2seq models are shown in Table 3. Clearly, seq2seq performs better than Shang et al. (2018), the reason being a seq2seq model trained on a sufficiently large dataset is better at summarization than the unsupervised sentence compression method. The lower performance of cluster→seq2seq in comparison to seq2seq shows not only that a disentanglement component is unnecessary but also illustrates the error propagation of disentanglement to summarization.

Model	Rouge-1	Rouge-2	Rouge-L
Shang et al. (2018)	30.37	10.77	20.04
seq2seq	**44.38**	**19.47**	**35.20**
cluster→seq2seq	42.93	18.76	30.68

Table 3: Rouge F1-Scores for seq2seq models on the Easy Corpus.

6 Seq2seq vs. hier2hier models

We then compare the three proposed hierarchical approaches against the seq-to-seq approach in summarizing the interleaved texts by experimenting on the Medium and Hard corpora. Table 4 shows the experimental results; clearly, an increase in the complexity of interleaving reduces the performances throughout the models, especially in Rouge-2 and Rouge-L by ≈ 0.5 points. Further, Table 4 also shows the change in performance with the change of encoder-decoder components starting from seq2seq. Evidently, a change of encoder from sequentially to hierarchically gives only a minor improvement, but the enhancement in the speed of training ($\approx 2\times$ with a Tesla V100 GPU) with this change is very essential, as often a sequential RNN model takes a week to converge. A noticeable improvement is observed on changing the decoder to hierarchical, i.e., ≈ 2 Rouge points. However, the difference in performance with the type of hierarchical decoding, i.e., threadLSTM or feedbackLSTM, is minor.

Model	Rouge-1	Rouge-2	Rouge-L
Medium Corpus			
seq2seq	38.78	16.47	30.12
hier2seq	39.42	16.18	29.99
hier2hier_tLSTM	**41.33**	17.10	**32.14**
hier2hier_fLSTM	40.83	**17.29**	31.73
Hard Corpus			
seq2seq	38.76	15.90	28.48
hier2seq	39.19	15.62	28.33
hier2hier_tLSTM	41.21	16.30	30.13
hier2hier_fLSTM	**41.76**	**16.89**	**30.70**

Table 4: Rouge F1-Scores of models on the Medium and Hard Corpus. hier2hier_tLSTM refers to hierarchical decoding using threadLSTM. hier2hier_fLSTM refers to hierarchical decoding using feedbackLSTM.

7 Hierarchical attention: with vs without

To understand the impact of hierarchical attention on the hierarchical decoders, we perform an ablation study of post-level attentions ($\boldsymbol{\beta}$), and use the Hard corpus for the experiments. The contribution of the post-level attentions in a hierarchical decoder is two-fold: computing the thread representation and re scaling the word-level attentions.

In threadLSTM, the thread-to-thread decoder, D_{t2t}, utilizes the post-level attention through its input; see Section 3.2.1. Also, depending on encoder type, the word-level attentions in threadLSTM are rescaled either using Eq. 5 or Eq. 7. In this ablation study, we assign 1 to $\boldsymbol{\beta}$ values, and thereby, changing the input of D_{t2t} to $\frac{1}{n}\sum_{j=1}^{n}\mathbf{m}_j$, and also word attention from $\hat{\boldsymbol{\alpha}}$ (Eq. 5 or Eq. 7) to default Bahdanau et al. (2014)'s $\boldsymbol{\alpha}$. In feedbackLSTM, the ablation changes remain the same

Model	HAttn.	Rouge-1	Rouge-2	Rouge-L
		Sequential Encoder		
tLSTM	No	39.67	15.32	29.00
	Yes	**40.83**	**15.98**	**29.81**
fLSTM	No	39.11	15.11	29.12
	Yes	**40.11**	**15.59**	**29.80**
		Hierarchical Encoder		
tLSTM	No	38.49	14.76	28.63
	Yes	**41.21**	**16.3**	**30.13**
fLSTM	No	38.45	14.77	28.21
	Yes	**41.76**	**16.89**	**30.70**

Table 5: Rouge F1-Score of models on the Hard Corpus. tLSTM refers to threadLSTM, fLSTM to feedbackLSTM and HAttn to hierarchical attention. Under Sequential Encoder, *LSTM refers to seq2hier_*LSTM. Under Hierarchical Encoder, *LSTM refers to hier2hier_*LSTM.

as the threadLSTM except the network k computing thread representation now takes a simple average ($\frac{1}{n}\sum_{j=1}^{n}\mathbf{m}_j$) as the input.

Table 5 shows the performance comparison. Clearly, models with hierarchical attention have better performance than ones without. Moreover, the enhancement in case of the hierarchical encoders is higher than the sequential ones as it integrates more appropriately to the hierarchical arrangement of encoder information.

8 Discussion

Occurrence of interleaved texts is common; however, readers often don't have the patience to wade through them. Surveys have shown that summarized contents are easy to consume, and thus, many organization have begun displaying extractive summaries, content visualization or the combination alongside the interleaved texts.[2] However, while an extractive system selects topic-wise high ranking posts, it may fail to capture the entire argument corresponding to them. Instead, we propose an end-to-end abstractive system which not only evades error propagation but also hyper-parameters tuning like thread size.

Table 6 shows an output of our hierarchical abstractive system, in which interleaved texts and generated multi-sentence summaries are in the top and bottom respectively. Table 6 also shows the top two indexes of the interleaved sentences that the post-level attention ($\boldsymbol{\beta}$) attends while generating those summaries. The system not only manages to disentangle the interleaved texts but also

[2]http://resources.trustyou.com/c/wp-present-travel-review-content?x=0MFT5U

	Interleaved Texts
0	to study the effect of long-term , self-monitored **exercise** on **physical** . . .
1	the objective was to evaluate the effect of high-flux **hemodialysis** on. . .
2	seventeen women were randomly allocated to jog 2h/wk for 4 months on. . .
3	. . . controlled trial was set up in 1985 to test the effect of **social**. . .
4	the study was double-blind single cross-over with random allocation to. . .
5	although leisure-time physical activity increased significantly in. . .
6	although the effect of **imagery** perspectives , i.e . the patients were. . .
⋮	⋮
13	both the conventional and high-flux membranes were cellulose acetate. . .
⋮	⋮
	GroundTruth
•	little effect of long-term , self-monitored exercise on serum lipid levels in middle-aged women .
•	effect of high-flux hemodialysis on quality of life and neuropsychological function in chronic hemodialysis patients .
•	social intervention and the elderly : a randomized controlled trial .
•	the effect of imagery perspectives on the psychophysiological responses to imagined exercise .
	Generation
•	effects of **physical** fitness and **exercise** on physical fitness and serum lipids of middle-aged females **0, 2**
•	[UNK] **hemodialysis** : a double-blind , placebo-controlled study . **1, 13**
•	the effect of **social** intervention on [UNK] physical activity in elderly people living with dementia . **5, 3**
•	effects of **imagery** on [UNK] responses to exercise . **6, 9**

Table 6: Interleaved sentences of 4 articles, ground-truth multi-sentence summary and hier2hier summary generation. Indices of the top 2 sentences that were attended ($\boldsymbol{\beta}$) for the generation are in bold.

to generate appropriate abstractive summaries. Meanwhile, $\boldsymbol{\beta}$ also provides explainability of the output.

As interleaving in the table includes abstracts from the same category, e.g. Physical Activities, the interleaving is complex and approximates the real-world conversations. Despite that, the end-to-end hierarchical system tackles the task to a large extent. The next, future stage in this research is transfer learning of the hierarchical system to a real world system.

9 Conclusion

We presented an end-to-end trainable hierarchical encoder-decoder architecture which implicitly disentangles interleaved texts and generates a multi-

sentence abstractive summary covering the text threads. Furthermore, the architecture addresses the error propagation issue that occurs in the two-step architectures. Our proposed novel hierarchical attention further boosts both disentanglement and summary generation.

References

Ahmet Aker, Monica Paramita, Emina Kurtic, Adam Funk, Emma Barker, Mark Hepple, and Rob Gaizauskas. 2016. Automatic label generation for news comment clusters. In *Proceedings of the 9th International Natural Language Generation Conference*, pages 61–69.

Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. 2012. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Emma Barker, Monica Lestari Paramita, Ahmet Aker, Emina Kurtic, Mark Hepple, and Robert Gaizauskas. 2016. The sensei annotated corpus: Human summaries of reader comment conversations in on-line news. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 42–52.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 93–98. The Association for Computational Linguistics.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313. Asian Federation of Natural Language Processing.

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

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. 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 1812–1822. Association for Computational Linguistics.

Baoyu Jing, Pengtao Xie, and Eric Xing. 2018. On the automatic generation of medical imaging reports. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2577–2586. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Patterm Recognition (CVPR)*.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115. Association for Computational Linguistics.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2012. Topic-driven reader comments summarization. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 265–274. ACM.

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

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

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

Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 379–389. The Association for Computational Linguistics.

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. Association for Computational Linguistics.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674. Association for Computational Linguistics.

Suzan Verberne, Emiel Krahmer, Iris Hendrickx, Sander Wubben, and Antal van Den Bosch. 2018. Creating a reference data set for the summarization of discussion forum threads. *Language Resources and Evaluation*, pages 1–23.

Lidan Wang and Douglas W Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 200–208. Association for Computational Linguistics.

Yuan Xue, Tao Xu, L. Rodney Long, Zhiyun Xue, Sameer K. Antani, George R. Thoma, and Xiaolei Huang. 2018. Multimodal recurrent model with attention for automated radiology report generation. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, volume 11070 of *Lecture Notes in Computer Science*, pages 457–466. Springer.