Coarse-to-Fine Attention Models for Document Summarization

Jeffrey Ling and **Alexander M. Rush**
Harvard University
{jling@college,srush@seas}.harvard.edu

Abstract

Sequence-to-sequence models with attention have been successful for a variety of NLP problems, but their speed does not scale well for tasks with long source sequences such as document summarization. We propose a novel coarse-to-fine attention model that hierarchically reads a document, using coarse attention to select top-level chunks of text and fine attention to read the words of the chosen chunks. While the computation for training standard attention models scales linearly with source sequence length, our method scales with the number of top-level chunks and can handle much longer sequences. Empirically, we find that while coarse-to-fine attention models lag behind state-of-the-art baselines, our method achieves the desired behavior of sparsely attending to subsets of the document for generation.

1 Introduction

The sequence-to-sequence architecture of Sutskever et al. (2014), also known as the encoder-decoder architecture, is now the gold standard for many NLP tasks, including machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), question answering (Hermann et al., 2015), dialogue (Li et al., 2016), caption generation (Xu et al., 2015), and in particular summarization (Rush et al., 2015).

A popular variant of sequence-to-sequence models are *attention* models (Bahdanau et al., 2015). By keeping an encoded representation of each part of the input, we "attend" to the relevant part each time we produce an output from the decoder. In practice, this means computing attention weights for all encoder hidden states, then taking the weighted average as our new context vector.

While successful, existing sequence-to-sequence methods are computationally limited by the length of source and target sequences. For a problem such as document summarization, a source sequence of length N (where N could potentially be very large) requires $O(N)$ model computations to encode. However, it makes sense intuitively that not every word of the source will be necessary for generating a summary, and so we would like to reduce the amount of computation performed on the source.

Therefore, in order to scale attention models for this problem, we aim to prune down the length of the source sequence in an intelligent way. Instead of naively attending to all the words of the source at once, our solution is to use a two-layer hierarchical attention. For document summarization, this means dividing the document into chunks of text, sparsely attending to one or a few chunks at a time using hard attention, then applying the usual full attention over those chunks – we call this method *coarse-to-fine attention*. Through experiments, we find that while coarse-to-fine attention does not perform as well as standard attention, it does show the desired behavior of sparsely reading the source sequence.

We structure the rest of the paper as follows. In Section 2, we introduce related work on summarization and neural attention. In Section 3, we review the encoder-decoder framework, and in Section 4 introduce our models. In Section 5, we describe our experimental setup, and in Section 6 show results. Finally, we conclude in Section 7.

2 Related Work

In summarization, neural attention models were first applied by Rush et al. (2015) to do headline

Proceedings of the Workshop on New Frontiers in Summarization, pages 33–42
Copenhagen, Denmark, September 7, 2017.

generation, i.e. produce a title for a news article given only the first sentence. Nallapati et al. (2016) and See et al. (2017) apply attention models to summarize full documents, achieving state-of-the-art results on the CNN/Dailymail dataset. All of these models, however, suffer from the inherent complexity of attention over the full document. Indeed, See et al. (2017) report that a single model takes over 3 days to train.

Many techniques have been proposed in the literature to efficiently handle the problem of large inputs to deep neural networks. One particular framework is that of "conditional computation", as coined by Bengio et al. (2013) — the idea is to only compute a subset of a network's units for a given input by gating different parts of the network.

Several methods, some stochastic and some deterministic, have been explored in the vein of conditional computation. In this work, we will focus on stochastic methods, although deterministic methods are worth considering as future work (Rae et al., 2016; Shazeer et al., 2017; Miller et al., 2016; Martins and Astudillo, 2016).

On the stochastic front, Xu et al. (2015) demonstrate the effectiveness of "hard" attention. While standard "soft" attention averages the representations of where the model attends to, hard attention discretely selects a single location. Hard attention has been successfully applied in various computer vision tasks (Mnih et al., 2014; Ba et al., 2015), but so far has limited usage in NLP. We will apply hard attention to the document summarization task by sparsifying our reading of the source text.

3 Background

We begin by describing the standard sequence-to-sequence attention model, also known as encoder-decoder models.

In the encoder-decoder architecture, an *encoder* recurrent neural network (RNN) reads the source sequence as input to produce the *context*, and a *decoder* RNN generates the output sequence using the context as input.

Formally, suppose we have a vocabulary $\mathcal{V}$. A given input sequence $w_1, \ldots, w_n \in \mathcal{V}$ is transformed into a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{d_{in}}$ through a word embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d_{in}}$ as $\mathbf{x}_t = \mathbf{E}w_t$.

The encoder RNN is given by a parameterizable function f_{enc} and a hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{hid}}$ at each time step t with $\mathbf{h}_t = f_{enc}(\mathbf{x}_t, \mathbf{h}_{t-1})$. In our models, we use the long-short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997).

The decoder is another RNN f_{dec} that generates output words $y_t \in \mathcal{V}$. It keeps hidden state $\mathbf{h}_t^{dec} \in \mathbb{R}^{d_{hid}}$ as $\mathbf{h}_t^{dec} = f_{dec}(y_t, \mathbf{h}_{t-1}^{dec})$ similar to the encoder RNN. A context vector is produced at each time step using an attention function a that takes the encoded hidden states $[\mathbf{h}_1, \ldots, \mathbf{h}_n]$ and the current decoder hidden state $\mathbf{h}_t^{dec}$ and produces the context $\mathbf{c}_t \in \mathbb{R}^{d_{ctx}}$: $\mathbf{c}_t = a([\mathbf{h}_1, \ldots, \mathbf{h}_n], \mathbf{h}_t^{dec})$. As in Luong et al. (2015), we feed the context vector at time $t-1$ back into the decoder RNN at time t, i.e. $\mathbf{h}_t^{dec} = f_{dec}([y_t, \mathbf{c}_{t-1}], \mathbf{h}_{t-1}^{dec})$.

Finally, a linear projection and softmax (the generator) produces a distribution over output words $y_t \in \mathcal{V}$:

$$p(y_t | y_{t-1}, \ldots, y_1, [\mathbf{h}_1, \ldots, \mathbf{h}_n]) = \\ \text{softmax}(\mathbf{W}^{out}\mathbf{c}_t + \mathbf{b}^{out})$$

The models are then trained end-to-end to minimize negative log-likelihood loss (NLL).

We note that we have great flexibility in how our attention function $a(\cdot)$ combines the encoder context and the current decoder hidden state. In the next section, we describe our models for $a(\cdot)$.

4 Models

We describe a few instantiations for the attention function $a(\cdot)$: **standard attention**, **hierarchical attention**, and **coarse-to-fine attention**.

4.1 Standard Attention

In Bahdanau et al. (2015), the function $a(\cdot)$ is implemented with an *attention network*. We compute attention weights for each encoder hidden state h_i as follows:

$$\beta_{t,i} = \mathbf{h}_i^\top \mathbf{W}^{attn} \mathbf{h}_t^{dec} \quad \forall i = 1, \ldots, n \qquad (1)$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{\beta}_t) \qquad (2)$$

$$\widetilde{\mathbf{c}}_t = \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i \qquad (3)$$

Attention allows us to select the most relevant words of the source (by assigning higher attention weights) when generating words at each time step.

Our final context vector is then $\mathbf{c}_t = \tanh(\mathbf{W}^2[\widetilde{\mathbf{c}}_t, \mathbf{h}_t^{dec}])$ for $\mathbf{W}^2 \in \mathbb{R}^{2d_{hid} \times d_{ctx}}$ a learned matrix.

Going forward, we call this instantiation of the attention function STANDARD.

4.2 Hierarchical Attention

The attention network of STANDARD is computationally expensive for long sequences — for each hidden state of the decoder, we need to compare it to every hidden state of the encoder in order to determine where to attend to. This seems unnecessary for a problem such as document summarization; intuitively, we only need to attend to a few important chunks of text at a time. Therefore, we propose a hierarchical method of attending to the document — by segmenting the document into large top-level chunks of text, we first attend to these chunks, then to the words within the chunks.

To accomplish this hierarchical attention, we construct encodings of the document at both levels. Suppose we have chunks $s_1, \ldots, s_m$ with words $w_{i,1}, \ldots, w_{i,n_i}$ in chunk s_i. For the top-level representations, we use a simple encoding model (e.g. bag of words or convolutions) on each s_i to obtain hidden states $\mathbf{h}^s_i \in \mathbb{R}^{d_{sent}}$ (see Section 5 for details). For the word representations, we run an LSTM encoder separately on the words of each chunk; specifically, we apply an RNN on s_i to get hidden states $\mathbf{h}_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n_i$ where $\mathbf{h}_{i,j} = \text{RNN}(\mathbf{h}_{i,j-1}, w_{i,j})$.

Using the top-level representations $\mathbf{h}^s_i$ and the word representations $\mathbf{h}_{i,j}$, we compute coarse attention weights $\alpha^s_1, \ldots, \alpha^s_m$ for the top-level chunks in the same way as STANDARD, and similarly compute fine attention weights $\alpha^w_{i,1}, \ldots, \alpha^w_{i,n_i}$ for each i. We then compute the final soft attention on word $w_{i,j}$ as $\alpha_{i,j} = \alpha^s_i \cdot \alpha^w_{i,j}$ (note this ensures that the weights normalize to 1 over the whole document). Finally, we proceed exactly as in standard attention by computing the weighted average over hidden states $\mathbf{h}_{i,j}$ to produce the context, i.e. $\widetilde{\mathbf{c}} = \sum_{i,j} \alpha_{i,j} \mathbf{h}_{i,j}$.

We label this attention method HIER. Next, we consider the hard attention version of this model to achieve sparsity in our network.

4.3 Coarse-to-Fine Attention

With the previous models STANDARD and HIER, we are required to compute hidden states over all words and top-level chunks in the document, so that if we have M chunks and N words per chunk, the computational complexity is $O(MN)$ for each attention step.

However, if we are able to perform conditional computation and only read M^+ of the chunks at a time, we can reduce the attention complexity to $O(M + M^+N)$, where we choose the chunks to attend to in $O(M)$ and read the selected chunks in $O(M^+N)$. Note that this expression ignores the total the number of words of the document, and the bottleneck becomes the length of each chunk of text.

In our model, we will apply stochastic sampling to the top-level attention distribution in the spirit of hard attention (Xu et al., 2015; Mnih et al., 2014; Ba et al., 2015) while keeping the lower-level attention as is. We call our method *coarse-to-fine attention*[1].

Specifically, using the top-level attention distribution $\alpha^s_1, \ldots, \alpha^s_m$, we select a single chunk s_i by sampling this distribution. We then set the context vector as $\sum_{j=1}^{n_i} \alpha^w_{i,j} \mathbf{h}_{i,j}$, where we use the word attention weights for the chosen chunk s_i. Note that this is equivalent to converting the top-level distribution α^s_i to a one-hot encoding based on the hard sample, then writing $\alpha_{i,j} = \alpha^s_i \cdot \alpha^w_{i,j}$ as in HIER. At test time, we take the max α^s_i for a one-hot encoding instead of sampling. We label this coarse-to-fine method C2F.

Because the hard attention model loses the property of being end-to-end differentiable, we use reinforcement learning to train our network. Specifically, we use the REINFORCE algorithm (Williams, 1992), also formalized by Schulman et al. (2015) in the stochastic computation graph framework. Layers before the hard attention node receive backpropagated policy gradient $\frac{\partial \mathcal{L}}{\partial \theta} = r \cdot \frac{\partial \log p(\alpha|\theta)}{\partial \theta}$, where r is some reward and $p(\alpha|\theta)$ is the attention distribution that we sample from.

Rewards and variance reduction We can think of our decoder RNN as a reinforcement learning agent where the state is the LSTM decoder state at time t and actions are the hard attention decisions. Since samples from $\boldsymbol{\alpha}_t$ at time t of the RNN decoder can also affect future rewards, the total influenced reward is $\sum_{s=t}^{T} r_s$ at time t, where $r_t = \log p(y_t|y_1, \ldots, y_{t-1}, \mathbf{x})$ is the single step reward. Inspired by the discount factor from RL, we slightly modify the total reward: instead of simply taking the sum, we can scale later rewards with a discount factor γ, giving total reward $\sum_{s=t}^{T} \gamma^{s-t} r_s$ for the stochastic hard attention node a_t. We found

[1]The term coarse-to-fine attention has previously been introduced in the literature (Mei et al., 2016). However, their idea is different: they use coarse attention to reweight the fine attention computed over the entire input. This idea has also been called hierarchical attention (Nallapati et al., 2016).

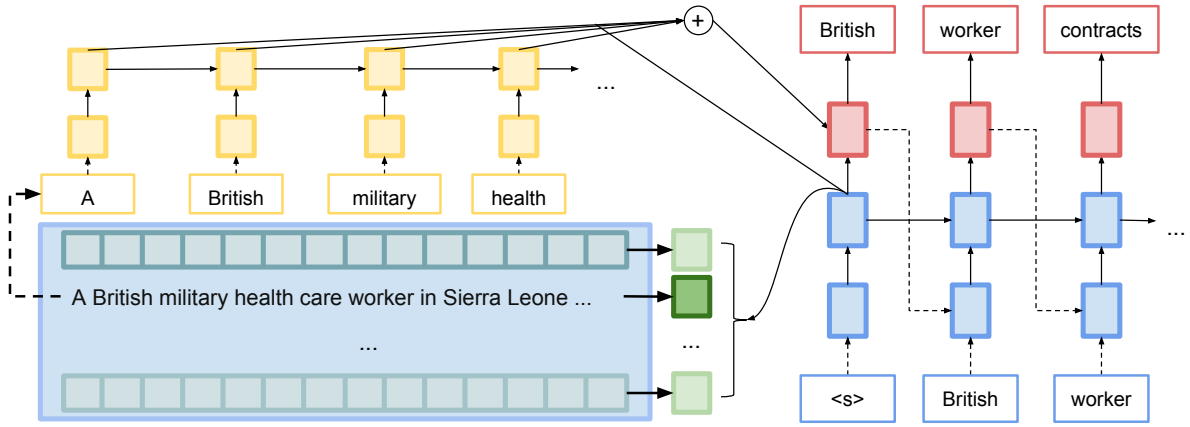

Figure 1: Model architecture for sequence-to-sequence with coarse-to-fine attention. The left side is the encoder that reads the document, and the right side is the decoder that produces the output sequence. On the encoder side, the top-level hidden states are used for the coarse attention weights, while the word-level hidden states are used for the fine attention weights. The context vector is then produced by a weighted average of the word-level states. In HIER, we average over the coarse attention weights, thus requiring computation of all word-level hidden states. In C2F, we make a hard decision for which chunk of text to use, and so we only need to compute word-level hidden states for one chunk.

that adding a discount factor helps in practice (we use $\gamma = 0.5$).

Training on the reward directly tends to have high variance, and so we subtract a baseline reward to help reduce variance as per Weaver and Tao (2001). To calculate these baselines, we store a constant b_t for each decoder time step t. We follow Xu et al. (2015) and keep an exponentially moving average of the reward for each time step t as $b_t \leftarrow b_t + \beta(r_t - b_t)$ where r_t is the average minibatch reward and β a learning rate (set to 0.1).

In addition to including a baseline, we also scale the rewards by a tuned hyperparameter λ — we found that scaling helped to stabilize training. We empirically set λ to 0.3. Therefore, our final reward at time t can be written as

$$\lambda \sum_{s=t}^{T} \gamma^{s-t}(r_s - b_s) \tag{4}$$

ALTERNATE training Xu et al. (2015) explain that training hard attention with REINFORCE has very high variance, even when including a baseline. Thus, for every minibatch of training, they randomly use soft attention instead of hard attention with some probability (they use 0.5). The backpropagated gradient is then the standard soft attention gradient instead of the REINFORCE gradient. When we use this training method in our results, we label it as +ALTERNATE.

Multiple samples From our initial experiments with C2F, we found that taking a single sample was not very effective. However, we discovered that sampling multiple times from the attention distribution $\boldsymbol{\alpha}^s$ improves performance.

To be precise, we fix a number k_{mul} for the number of times we sample from $\boldsymbol{\alpha}^s$. Then, we sample based on the multinomial distribution $\boldsymbol{\mu} \sim \text{Mult}(k_{mul}, \{\alpha_i\}_{i=1}^m)$ to produce the new top-level attention vector $\widetilde{\boldsymbol{\alpha}}^s$, with $\widetilde{\alpha}_i^s = \mu_i / k_{mul}$. In our results, we label this as +MULTI.

Intuitively, k_{mul} is the number of top-level chunks we select to produce the context. With higher k_{mul}, the hard attention model more closely approximates the soft attention model, and hence should lead to better performance. This, however, incurs a cost in computational complexity.

5 Experiments

5.1 Data

Experiments were performed on a version of the CNN/Dailymail dataset from Hermann et al. (2015). Each data point is a news document accompanied by up to 4 "highlights", and we take the first of these as our target summary. Note that our dataset differs from related work (Nallapati et al., 2016; See et al., 2017) which take all the highlights as the summary, as we were less interested in target side length and more in correctly locating sparse attention in the source.

Train, validation, and test splits are provided with the original dataset along with document tokenization and sentence splitting. We do addi-

tional preprocessing by replacing all numbers with # and appending end of sentence tokens </s> to each sentence. We limit our vocabulary size to the 50000 most frequent words, replacing the rest with <unk> tokens.

5.2 Implementation Details

To ease minibatch training on the hierarchical models, we arrange the first 400 words of the document into a 10 by 40 image and take each row to be a top-level chunk. For HIER, we also experiment with shapes of 5 by 80 and 2 by 200 (denoted 5X80, 2X200 resp.). These should more closely approximate STANDARD as the shape approaches a single sequence.

In addition, we pad short documents to the maximum length with a special padding word and allow the model to attend to it. However, we zero out word embeddings for the padding states and also zero out their corresponding LSTM states. We found in practice that very little of the attention ended up on the corresponding states.

5.3 Models

Baselines We consider a few baseline models. A strong and simple baseline is the first sentence of the document, which we denote FIRST.

We also consider the integer linear programming (ILP) based document summarizer of Durrett et al. (2016). We apply the code [2] directly on the test set without retraining the system. We provide the necessary preprocessing using the Berkeley coreference system[3]. We call this baseline ILP.

Our models We ran experiments with the models STANDARD, HIER, and C2F as described above.

For the coarse attention representations $\mathbf{h}_i^s$ of HIER and C2F, we experiment with convolutional and bag of words encodings. We use convolutions for the top-level representations by default, where we follow Kim (2014) and perform a convolution over each window of words in the chunk using 600 filters of kernel width 6. We use max-over-time pooling to obtain a fixed-dimensional top-level representation in $\mathbb{R}^{d_f}$ where $d_f = 600$ is the number of filters. For bag of words, we simply take the top-level representation as the sum of the chunk's word embeddings (for a separate embedding matrix), and we write BOW when we use this encoding. For BOW models, we fix the word embeddings on the encoder side (in other models, they are fine tuned).

As an addition to any top-level representation method, we can include *positional embeddings*. In general, we expect the order of text in the document to matter for summarization — for example, the first few sentences are usually important. We therefore include the option to concatenate a 25-dimensional embedding of the chunk's position to the representation. When we use positional embeddings, we write +POS.

For C2F, we include options +MULTI for $k_{mul} > 1$, +PRETRAIN for starting with a model pretrained with soft attention for 1 epoch, and +ALTERNATE for sampling between hard and soft attention with probability 0.5.

5.4 Training

We train with minibatch stochastic gradient descent (SGD) with batch size 20 for 20 epochs, renormalizing gradients below norm 5. We initialize the learning rate to 0.1 for the top-level encoder and 1 for the rest of the model, and begin decaying it by a factor of 0.5 each epoch after the validation perplexity stops decreasing.

We use 2 layer LSTMs with 500 hidden units, and we initialize word embeddings with 300-dimensional word2vec embeddings (Mikolov and Dean, 2013). We initialize all other parameters as uniform in the interval $[-0.1, 0.1]$. For convolutional layers, we use a kernel width of 6 and 600 filters. Positional embeddings have dimension 25. We use dropout (Srivastava et al., 2014) between stacked LSTM hidden states and before the final word generator layer to regularize (with dropout probability 0.3). At test time, we run beam search to produce the summary with a beam size of 5.

Our models are implemented using Torch based on a past version of the OpenNMT system[4] (Klein et al., 2017). We ran our experiments on a 12GB Geforce GTX Titan X GPU. The models take between 2-2.5 hours to train per epoch.

5.5 Evaluation

We report metrics for perplexity and ROUGE balanced F-scores (Lin, 2004) on the test set.

[2] https://github.com/gregdurrett/berkeley-doc-summarizer
[3] https://github.com/gregdurrett/berkeley-entity
[4] http://opennmt.net

With multiple gold summaries in the CNN/Dailymail highlights, we take the max ROUGE score over the gold summaries for a predicted summary, as our models are trained to produce a single sentence. The final metric is then the average over all test data points.[5]

Note that because we are training the model to output a single highlight, our numbers are not comparable with Nallapati et al. (2016) or See et al. (2017).

6 Results

Table 1 shows summarization results. We see that our soft attention models comfortably beat the baselines, while hard attention lags behind.

The ILP model ROUGE scores are surprisingly low. We attribute this to the fact that our models usually produce a single sentence as the summary, while the ILP system can produce multiple. ILP therefore has comparatively high ROUGE recall while suffering in precision.

Unfortunately, the STANDARD sequence-to-sequence baseline proves to be difficult to beat. HIER performs surprisingly poorly, even though the hierarchical assumption seems like a natural one to make. We believe that the assumption that we can factor the attention distribution into learned coarse and fine factors may in fact be too strong. Because the training signal is back-propagated to the word-level LSTM via the coarse attention, the training algorithm cannot directly compare word attention weights as in STANDARD. Thus, the model does not learn how to attend to the most relevant top-level chunks, instead averaging the attention as a backoff (see 6.1). Additionally, the shapes 5X80 and 2X200 perform slightly better, indicating that the model prefers to have fewer sequences to attend to.

C2F results are significantly worse than soft attention results. As has been previously observed (Zaremba and Sutskever, 2015), training with reinforcement learning is inherently more difficult than standard maximum likelihood, as the signal from rewards tends to have high variance (even with variance reduction techniques). Thus, it may be too difficult to train the encoder (which forms a large part of the model) using such a noisy gradient. Even with soft attention pretraining (+PRETRAIN) and alternating training (+ALTERNATE), C2F fails to reach HIER performance.

While taking a single sample performs quite poorly, we see that taking more than one sample gives a significant boost to scores (+MULTI2, +MULTI3). There seem to be diminishing returns as we take more samples.

Finally, we note that positional embeddings (+POS) give a nontrivial boost to scores and causes the attention to prefer the front of the document. The exception, C2F + POS, is due to the fact that the attention collapses to always highlight the first top-level chunk.

We show predicted summaries from each model in Figure 2. We note that the ILP system, which extracts sentences first, produces long summaries. In contrast, the generated summaries tend to be quite succint, and most are the result of copying or paraphrasing specific sentences.

Source: isis supporters have vowed to murder twitter staff because they believe the site 's policy of shutting down their extremist pages is a ' virtual war ' . </s> a mocked - up image of the site 's founder jack dorsey in <unk> was posted yesterday alongside a diatribe written in arabic , which claimed twitter employees ' necks are ' a target for the soldiers of the caliphate ' . </s> addressing mr dorsey personally , it claimed twitter was taking sides in a ' media war ' which allowed ' slaughter ' , adding : ' your virtual war on us will cause a real war on you . </s> diatribe : an image of twitter founder jack dorsey in <unk> was posted alongside a rant in arabic </s> ...

GOLD: diatribe in arabic posted anonymously yesterday and shared online
FIRST: isis supporters have vowed to murder twitter staff because they believe the site 's policy of shutting down their extremist pages is a ' virtual war ' .
ILP: ISIS supporters have vowed to murder Twitter staff because they believe the site 's policy of shutting down their extremist pages is a ' virtual war ' . Twitter was taking sides . Islamic State militants have swept through huge tracts of Syria and Iraq , murdering thousands of people .

STANDARD: image of jack dorsey 's founder jack dorsey posted on twitter
HIER: the message was posted in arabic and posted on twitter
HIER BOW: the message was posted on twitter and posted on twitter
HIER +POS: dorsey in <unk> was posted yesterday alongside a diatribe in arabic
C2F: ' lone war ' is a ' virtual war ' image of the islamic state
C2F +MULTI2: isis supporters say site 's policy of shutting down is a ' propaganda war '
C2F +POS +MULTI2: twitter users say they believe site 's policy of closure is a ' media war '

Figure 2: Predicted summaries for each model. The source document is truncated for clarity.

6.1 Analysis

Sharpness of Attention We are interested in measuring the ability of our models to focus on a single top-level chunk using attention. Quantitatively, we measure the entropy of the coarse attention on the validation set in Table 2. Intu-

[5] We run the ROUGE 1.5.5 script with flags `-m -n 2 -a -f B`.

Model	PPL	ROUGE-1	ROUGE-2	ROUGE-L
FIRST	-	32.3	15.5	27.4
ILP	-	29.1	16.0	26.5
STANDARD	13.9	34.7	18.8	32.3
HIER	16.0	33.3	17.5	31.0
HIER BOW	16.3	33.0	17.4	30.7
HIER +POS	15.4	34.2	18.3	31.8
HIER 5X80	15.0	33.9	18.0	31.5
HIER 2X200	14.5	33.9	18.1	31.6
C2F	32.8	28.2	12.9	26.2
C2F +POS	37.8	28.3	12.5	26.1
C2F +MULTI2	25.5	30.0	14.4	27.9
C2F +POS +MULTI2	21.9	31.2	15.3	29.0
C2F +MULTI3	22.9	30.4	14.9	28.3
C2F +PRETRAIN	26.3	29.7	14.2	27.5
C2F +ALTERNATE	23.6	31.1	15.4	28.8

Table 1: Summarization results for CNN/Dailymail (first highlight as target) on perplexity (PPL) and ROUGE metrics.

Model	Entropy
STANDARD	1.31
HIER	2.14
C2F	0.15
C2F +MULTI2	0.59
C2F +POS +MULTI2	0.46

Table 2: Entropy over coarse attention, averaged over all attention distributions in the validation set. For reference, uniform attention in our case gives entropy ≈ 2.30.

itively, higher entropy means the attention is more spread out, while lower entropy means the attention is concentrated.

We compute the entropy numbers by averaging over all generated words in the validation set. Because each document has been split into 10 chunks, perfectly uniform entropy would be ≈ 2.30.

We note that the entropy of C2F is very low (before taking the argmax at test time). This is exactly what we had hoped for — we will see that the model in fact learns to focus on only a few top-level chunks of the document over the course of generation. If we have multiple samples with +MULTI2, the model is allowed to use 2 chunks at a time, which relaxes the entropy slightly.

We also observe that the HIER entropy is very high and almost uniform. The model appears to be averaging the encoder hidden states across chunks, indicating that the training failed to find the same optimum as in STANDARD. We discuss this further in the next section.

Attention Heatmaps For the document in Figure 2, we visualize the coarse attention distributions produced by each model in Figure 3.

In each figure, the rows are the top-level chunks of each document (40 words per row), and the columns are the summary words produced by the model. The intensity of each box for a given column represents the strength of the attention weight on that row. For STANDARD, the heatmap is produced by summing the word-level attention weights in each row.

In HIER, we observe that the attention becomes washed out (in accord with its high entropy) and is essentially averaging all of the encoder hidden states. This is surprising because in theory, HIER should be able to replicate the same attention distribution as STANDARD.

If we examine the word-level attention (not pictured here), we find that the model focuses on stop words (e.g. punctuation marks, </s>) in the encoder. We posit this may be due to the LSTM "saving" information at these words, and so the soft attention model can best retrieve the information by averaging over these hidden states. Alternatively, the model may be ignoring the encoder and generating only from the decoder language model.

In C2F, we see that we get very sharp attention on some rows as we had hoped. Unfortunately, the model has trouble deciding where to attend to, oscillating between the first and second-to-last rows. We partially alleviate this problem by allowing the model to attend to multiple rows in hard attention. Indeed, with +MULTI2 +POS, the model actually produces a very coherent output by focusing attention near the beginning. We believe that the improved result for this example is not only due to more flexibility in where to attend, but a better

Figure 3: Sentence attention visualizations for different models. From left to right: (1) STANDARD, (2) HIER, (3) C2F, (4) C2F +MULTI2 +POS.

encoding model due to the training process.

7 Conclusion

In this work, we experiment with a novel coarse-to-fine attention model on the CNN/Dailymail dataset. We find that both versions of our model, HIER and C2F, fail to beat the standard sequence-to-sequence model on metrics, but C2F has the desired property of sharp attention on a small subset of the source. Therefore, coarse-to-fine attention shows promise for scaling up existing models to larger inputs.

Further experimentation is needed to improve these attention models to state of the art. In particular, we need to better understand (1) the reason for the subpar performance and high entropy of hierarchical attention, (2) how to control the variance training of reinforcement learning, and (3) how to balance the tradeoff between stronger models and attention sparsity over long source sequences. We would also like to investigate alternatives to reinforcement learning for implementing sparse attention, e.g. sparsemax (Martins and Astudillo, 2016) and key-value memory networks (Miller et al., 2016) (preliminary investigations with sparsemax were not extremely promising, but we leave this to future work). Resolving these issues can allow attention models to become more scalable, especially in computationally intensive tasks such as document summarization.

References

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple Object Recognition with Visual Attention. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation By Jointly Learning To Align and Translate. *ICLR* .

Yoshua Bengio, Nicholas Léonard, and Aaron C Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* pages 1998–2008.

KM Hermann, T Kocisky, and E Grefenstette. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems* pages 1–9.

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

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)* pages 1746–1751.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. *arXiv preprint arXiv:1603.06155* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *Emnlp* (September):11.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. *Proceedings of The 33rd International Conference on Machine Learning* pages 1614–1623.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. *Proceedings of NAACL-HLT* pages 1–11.

Tomas Mikolov and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems* .

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)* abs/1606.0:1400–1409.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray Kavukcuoglu. 2014. Recurrent models of visual attention. *Advances in Neural Information Processing Systems* pages 2204—-2212.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Proceedings of CoNLL* abs/1602.0:280–290.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. 2016. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 3621–3629.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* .

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient Estimation Using Stochastic Computation Graphs. *NIPS* pages 1–13.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks .

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: the Sparsely-Gated Mixture-of-Experts Layer. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958.

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

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence* pages 538–545.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML* 14:77—-81.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement Learning Neural Turing Machines. *CoRR* abs/1505.0.