Blending approaches to Abstractive Summarization

David Dowey
ddowey@stanford.edu

Abstract

Abstractive summarization with neural sequence-to-sequence techniques does a good job of making fluent, readable summaries, but can end-up including words and phrases which a human summarizer would not typically regard as the essential content. They can also incorrectly relay factual content and have a tendency to repeat themselves. In this paper, we take two state-of-the-art techniques to address these problems, the *bottom-up* attention mechanism and the *pointer-generator* network, and improve upon how they work together to achieve a sweet-spot between extractive and abstractive summarizers. We find that adding intra-attention on past states of the decoder to the pointer-generator architecture has the potential to improve performance. We also show the benefits of incorporating the prediction probabilities from the content-selector directly in the "copy-attention" mechanism, to improve the *bottom-up* approach.

1 Introduction

There are two distinct methods of neural summarization – extractive methods, which build summaries from selecting and compressing from the input text sequences – and abstractive methods which build summaries more like a human would, by creating new text passages containing words selected from a wider vocabulary. Much of the work in the last three years in abstractive summarization has focussed on variations in the attention mechanism to improve the selection of content, so that the abstraction in the summary contains the most important/relevant facts in the input sentences. Sequence-to-sequence (seq2seq) models with attention, through a pointer-generator mechanism, provide an effective way to copy words from source to summary, while novel words can still be generated from the decoder model See et al. [2017]. By contrast, extractive methods rely on neural models for the deletion of low-relevance passages and words, and then compress the remaining words to assemble the summary sentences. In bottom-up attention, the content selector does much the same, by masking content in the source document so that the copy attention in the pointer-generator mechanism attends to only the most salient parts Gehrmann et al. [2018].

In this paper, we focus on finding a better sweet-spot between extractive and abstractive models, by providing two extensions to the current state-of-the-art models. First, we make improvements to the attention mechanism in the pointer-generator model, by allowing the decoder contextual model and the probability governing when to copy words to attend to both the encoder states and the previous decoder states. Secondly, we alter how the content selection model is used to influence the choice of words to copy in the pointer-generator mechanism. The selection of content from the source document takes place within a two-step process, in contrast to the end-to-end process seen in many other models. The first step produces a selection mask to determine which most-relevant words and phrases can be used in the second step, which is the pointer-generator seq2seq model with intra-attention.

Since there are two distinct steps, both content selection and pointer-generator models can be trained separately. Therefore, the extensions we propose can be experimented with independently in order to tweak performance. We experimented with improvements to the attention architecture in the pointer-generator seq2seq model, so that the summary generated up to any time step can also also directly influence the choice of whether to copy or generate. Then we examined alternatives for

integrating the output from the content-selector model into the pointer-generator model in order to improve upon the pointer mechanism's ability to focus on the most salient words and phrases. The content-selector is itself a powerful extractive summarizer, and it is beneficial to incorporate its prediction probabilities directly in the pointer-generator's "copy-attention".

2 Our approach

We address the NLP task of summarization for which the goal is, given a source document, to produce a shorter version which keeps most of the original's meaning. Formally, we have article-summary pairs of text $(\mathbf{X}, \mathbf{Y})$, with $x_1, x_2, \ldots, x_n \in \mathbf{X}$ the individual tokens of the source article and $y_1, y_2, \ldots, y_m \in \mathbf{Y}$ the tokens of the summary, where the length of the summary is substantively shorter than the source $m << n$. The objective is to find a function $f(x : \theta)$ where θ is a set of learnable parameters that maximize the likelihood the model generating the correct sequences $y_1, y_2, \ldots, y_m$.

We build upon the basis of the pointer-generator seq2seq architecture, in which multi-sentence summaries are constructed by abstracting from the source text, generating the summary one word at a time at each time step. The sequence of input tokens x_j is fed, one-by-one, into a bi-directional LSTM encoder $\{\text{LSTM}^{e_{fwd}}, \text{LSTM}^{e_{bwd}}\}$ which computes a sequence of hidden states $\boldsymbol{h}_j^e = [\boldsymbol{h}_j^{e_{fwd}}; \boldsymbol{h}_j^{e_{bwd}}]$ from the embedding vectors of the x_j.

A uni-directional decoder LSTM^d computes a sequence of hidden states, $\boldsymbol{h}_t^d$, one-by-one, by concatenating the embedding vectors of the target summary sequences y_t with an *input feed* from the fused attentional context vector from the output of the previous time step, $\tilde{\boldsymbol{h}}_{t-1}^d$. The *input feed* mechanism allows the encoder to take into account the previous temporal alignments with the source words and target summary words, and therefore acts as an implicit *coverage* mechanism to avoid repeated words and phrases.

2.1 The attention architecture of the pointer-generator seq2seq model

Our main focus to more effectively model how the attention architecture is used to perform three tasks: to generate abstractive (new) words; to choose which words to extract from the source; and to determine whether to copy or generate. To motivate our extensions to the existing models, let us consider how a human approaches the summarization task.

When trying to summarize a longer article, a human first localizes the source content that is most relevant, in effect, this corresponds to how extractive summarizers and bottom-up attention approach the problem. It is perhaps key to why the bottom-up approach improves on other models. But, consider what happens next when human has started to summarize and has, say, finished creating half of the summary sentences. At this point, there is much less content of the original which is relevant for the completion of the summary. Focus narrows. Now, the selection of content for both the "copy-attention" and for the context vector used in the generation of new words would benefit from being more focussed on a smaller portion of the summary content.

In order to achieve a similar narrowing of focus as the summary progresses, we allow the uni-directional LSTM^d decoder to attend to its own previous outputs. This idea of double intra-attention was first proposed by Paulus et al. [2017], but we introduce it here in the framework of a bottom-up attention model. At each time step of the decoder, the context vector $\boldsymbol{c}_t^e$ captures the relevant source content on the encoder side. The encoder attention score is calculated between the hidden source states $\boldsymbol{h}^e$ of the encoder LSTM^e and the current hidden target state $\boldsymbol{h}_t^d$ of the decoder LSTM^d. A second context vector $\boldsymbol{c}_t^d$ captures the summary content on the decoder side which has been built-up in previous time steps. The decoder attention score is calculated between the hidden target states $\boldsymbol{h}_{0:t-1}^d$ and the current hidden target state $\boldsymbol{h}_t^d$.

At time step t for input word j, the pair of context vectors is given by:
Encoder:

$$e_{tj}^e = \text{score}\ (h_t^d, h_j^e)\ , \quad a_{tj}^e = \text{align}\ (h_t^d, h_j^e) = \frac{\exp(\text{score}\ (h_t^d, h_j^e)}{\sum_{j'} \exp(\text{score}\ (h_t^d, h_{j'}^e)}\ , \quad c_t^e = \sum_{j=1}^{n} a_{tj}^e\ h_j^e$$

Decoder:

$$e^d_{tt'} = \text{score}\ (h^d_t, h^d_{t'})\ , \quad a^d_{tt'} = \text{align}\ (h^d_t, h^d_{t'}) = \frac{\exp(\text{score}\ (h^d_t, h^d_{t'})}{\sum_{i=1}^{t-1} \exp(\text{score}\ (h^d_t, h^d_i)}\ , \quad c^d_t = \sum_{i=1}^{t-1} a^d_{tj}\ h^d_j$$

We then take the two context vectors and concatenate them, along with the current hidden target state $\boldsymbol{h}^d_t$ of the decoder LSTM^d to produce a fused attention state vector $\tilde{\boldsymbol{h}}^d_t$ in which the seq2seq representation contains information not only from the decoder $\boldsymbol{h}^d_t$ but also attends to information in the source document and the summary so far, through $\boldsymbol{c}^e_t$ and $\boldsymbol{c}^d_t$:

$$\tilde{\boldsymbol{h}}^d_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}^e_t; \boldsymbol{c}^d_t; \boldsymbol{h}^d_t])$$

To generate predictive probabilities on the target vocabulary, the fused attentional vector is passed through a softmax layer:

$$\text{P}_{\text{vocab}} = p(y_t|x, y_{1:t-1}) = \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{h}}^d_t)$$

Lastly, we consider two alternatives, multiplicative and additive, for the score function which produces the attention score vectors e^e_{tj} and $e^d_{tt'}$:

$$\text{score}(\boldsymbol{h}^d_t, \boldsymbol{h}^e_j) = \begin{cases} \boldsymbol{h}^{d\top}_t \boldsymbol{W}^e \boldsymbol{h}^e_j \\ \boldsymbol{v}^{e\top} \tanh(\boldsymbol{W}^e_1 \boldsymbol{h}^d_t + \boldsymbol{W}^e_2 \boldsymbol{h}^e_j) \end{cases} \qquad \text{score}(\boldsymbol{h}^d_t, \boldsymbol{h}^d_{0:t-1}) = \begin{cases} \boldsymbol{h}^{d\top}_t \boldsymbol{W}^{\boldsymbol{d}} \boldsymbol{h}^{\boldsymbol{d}}_{\boldsymbol{0:t-1}} \\ \boldsymbol{v}^{d\top} \tanh(\boldsymbol{W}^d_1 \boldsymbol{h}^d_t + \boldsymbol{W}^d_2 \boldsymbol{h}^d_{0:t-1}) \end{cases}$$

We experiment with these to see which performs better, applying the same score function to both the encoder and decoder in each experiment.

2.2 Choosing to copy or generate using the pointer-generator mechanism

We wish to make use of all the information available at time step t to make the choice whether to copy a word from the source document or to generate a word from the vocabulary. In contrast to See et al. [2017] where the *generation probability* was built from a sigmoid function of a representation including the decoder input, decoder state and the context vector, we prefer to calculate a *copy probability* directly from a sigmoid function of the fused attention vector. The problem is to predict a soft switch u_t at each time step that determines whether to copy ($u_t = 1$) an input word chosen through the copy-attention distribution or to generate ($u_t = 0$) a word from the vocabulary using P_{vocab}. Apart from being a simple solution, we believe that the fused attention vector already contains all the information (both context vectors and the decoder hidden state) needed to make the choice whether to copy or generate:

$$p_{\text{copy}} = \text{p}(u_t|x, y_{1:t-1}) = \sigma(\boldsymbol{W}_u\tilde{\boldsymbol{h}}^d_t + b_u)$$

In the bottom-up model proposed by Gehrmann et al. [2018] the linkage between the content-selection model probabilities and the pointer-generator network's attention distribution is through a hard threshold mask at evaluation time. The objective of the content selection model is to learn the probability q_j that word y_j is included in the summary. The mask on the source words corresponds to keeping only those $q_j > \epsilon$, where ϵ is some threshold value. The copy attention from the encoder at timestep t for word y_j is a^e_{tj}, and the copy attention distribution $\text{p}(a^e_t|x, y_{1:t-1})$ is used to determine which source words to copy. At the inference stage (only), content selection is applied as a hard mask to the copy attention distribution:

$$p(\tilde{a}^e_t|x, y_{1:t-1}) = \begin{cases} p(a^e_t|x, y_{1:t-1}), & \text{if } q_j > \epsilon \\ 0, & \text{otherwise} \end{cases}$$

This masking produces reduced set of source words to potentially copy to the summary.

We find this to be sub-optimal in several aspects. First, the mask is determined entirely by the content selection model predictions and a choice of cut-off threshold. Therefore, even in cases where the pointer-generator model has a strong copy-attention toward a source word, this high probability can be throttled entirely by the bottom-up mask. This is inconsistent with our motivation for a model in which the copy-attention adjusts as the summary progresses due to attending to the past decoder states. Rather than hard threshold, we multiply with copy-attention probabilities by those of the content-selection predictions:

$$p(\tilde{a}^e_t|x, y_{1:t-1}) = p(a^e_t|x, y_{1:t-1}) \times q_j$$

The resulting copy attention distribution is then normalized. Now, the choice of source words to potentially copy is not simply a reduced set, but rather the soft mask that accentuates or attenuates the probabilities from the pointer-generator model.

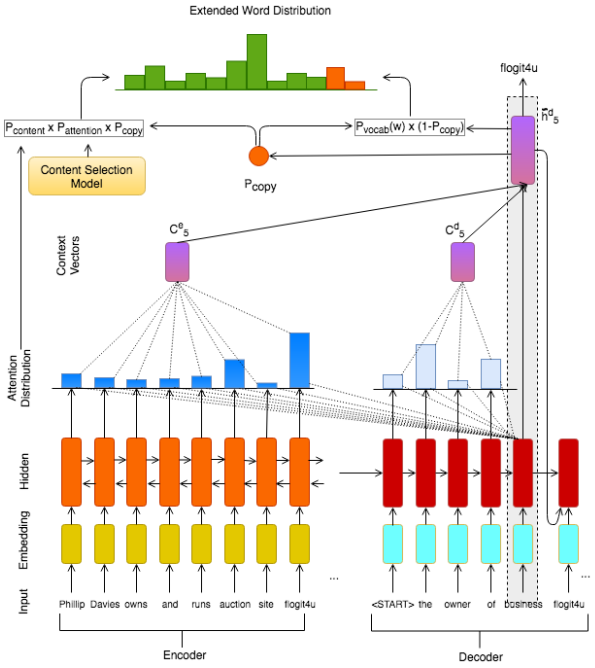
A second reason to prefer a soft mask, is that in experimenting with cut-off values the threshold, we have found that the bottom-up mechanism quickly goes from being highly extractive (threshold $\epsilon > 2$), with a small proportion of new words in the summary, to having little effect (threshold $\epsilon < 1$) and creates a summary entirely through the pointer-generator model. Our change, which combines content-selector predictions with the copy-attention distribution, allows a richer interaction between the bottom-up model and the summary generation at inference time.

Lastly, the authors of the bottom-up paper offer no guidance as to how to choose the threshold – it is truly an ad-hoc parameter, as it is sandwiched between the two stages of the model, and can only be tuned through experimentation. Since our intention for future work is to develop an end-to-end model, we would prefer a fully differentiable model where the mask probability is found by jointly optimizing the selection objective with the summarization objective. Therefore, our multiplicative probability solution is preferred to the hard threshold mask.

The resulting probability distribution of the pointer-generator network for the choice of word y is:

$$\mathrm{P}(y) = p_{\text{copy}} \times p(a_t^e | x, y_{1:t-1}) \times q_j + (1 - p_{\text{copy}}) \times \mathrm{P}_{\text{vocab}}(y)$$

Note that a_t^e is calculated from the sum of attentions of all the occurrences of word y in the source text. If the word y is not present in the source document, the first part will zero, and the model will choose entirely from $\mathrm{P}_{\text{vocab}}(y)$ – in contrast, for an OOV word, the latter part will be zero an the equation will be just the probability that the model will choose y and add it the extended vocabulary.



ORIGINAL TEXT
Nurses forced to work 12-hour shifts amid staff shortages are treated like 'slave labour' , a union has claimed. with a growing demand for nurses to work long hours, unison has called for more research into the effects of workers' health. At the union 's healthcare conference in Liverpool this week, one member highlighted how staff at a hospital in Staffordshire are expected to work six long days a week, totalling up to 78 hours a week. Jane Smith told colleagues : 'the fact staff are so tired has a detrimental effect on patient care and also puts other staff at risk. Nurses working 12-hour shifts amid staffing shortages in the NHS are treated like ' slave labour', one unison member has warned, as she called for more research into how longer working hours affect staff health. She said she was aware of one nurse who had worked 11 consecutive long day shifts, the nursing times reported. In a stretched NHS, longer shifts are now more common, offering a cost-effective strategy. [truncated]
REFERENCE SUMMARY
Unison member said tired staff have 'detrimental effect on patient care', Jane smith likened longer working hours for NHS staff to 'slave labour' called for more research to examine the impact on workers' health.
NEURAL MODEL SUMMARY
Unison has called for more research into the effects of workers' health. At the union 's healthcare conference in Liverpool this week , one member highlighted how staff at a hospital in Staffordshire are expected to work six long days a week . In a stretched NHS , longer shifts are now more common, offering a cost-effective strategy.

Figure 1: The pointer-generator model with context vectors from both the encoder and the decoder and prediction probabilities from the content selector. The summarization task seeks to produce a shorter version of the original source text which keeps most of the original's meaning.

The input feeding approach and keeping track of a coverage vector are two ways to avoid the repetition of words and phrases. Through input feeding, the fused attention state vector $\tilde{\boldsymbol{h}}_t^d$ is concatenated with the inputs at each time step. Through the state vector, the model is aware of previous alignments to the source inputs, but also the elements of the summary so far. This reduces the likelihood of repetition. We also consider model where a coverage vector cov_t is maintained, which is the sum of the encoder attention distributions a_t^e over all the previous time steps $cov_t = \sum_{t=0}^{t-1} a_t^e$. In previous time steps, source words that been attended to frequently will have their attention scores accumulate in the coverage vector, which will result in them being attended to less in future time steps. The full model is presented in Figure 1.

2.3 Related work

Intra-attention on both the encoder and the decoder in a pointer-generator mechanism was introduced by Paulus et al. [2017] in the framework of both a Maximum Likelihood model and a deep Reinforcement Learning model. We follow their approach to constructing the attention vectors and pointer-generator probabilities. Much more recent work by the same group Kryscinski et al. [2018] continues in the same spirit, but incorporates a pretrained language model in the decoder network to retrieve key elements of the source document. Their model, improves slightly the ROUGE score, but moreover produces more abstractive and novel summaries. The authors also discuss the ROUGE vs novelty trade-off.

The input feed approach was proposed by Luong et al. [2015] along with the core formulations for the attention score functions that we have used in this paper. Joining an extractor and abstractor in a single end-to-end model was the focus of recent paper Hsu et al. [2018] which also uses a dual level word – sentence attention on the decoder, for which consistency bewteen the two levels is ensured through an inconsistency loss function.

2.4 Methods and code

The baseline model architecture for the pointer-generator model used is the same as in See et al. [2017], with a single layer bi-directional LSTM with 128-dimensional word embeddings, 256 hidden states, no form of regularization, trained with Adagrad with 0.15 learning rate. Gradient-clipping was used with a maximum gradient norm of 2. As a baseline, the score function used for both the encoder and decoder attention is the additive attention vector. In contrast to the See et al. [2017] paper, for the baseline model we did not include a coverage vector, but rather used the input feed mechanism to reduce repetition. The inclusion of a specific coverage vector is one aspect that we wanted to examine in the experimentation.

Following Gehrmann et al. [2018] we apply penalties at inference and set a minimum summary length of 35 words to avoid incomplete summaries. To spur the model to generate longer sequences we use a length penalty as proposed in Wu et al. [2016] to normalize the loss function at inference time. The penalty lp is computed with the hyperparameter α adjusting the penalty factor through the function $lp(y) = \frac{(5+|y|)^\alpha}{(5+1)?\alpha}$, with larger α producing longer summaries.

For the first stage of content selection, we use the model presented in Gehrmann et al. [2018] which takes pre-trained GloVe word embeddings of size 100 and contextual ELMo embeddings of size 1024 (character token embeddings followed by two bi-directional LSTM layers) and then fine-tunes with a bi-directional LSTM layer for the prediction. The output of the model is a probability of inclusion for each source document word (the q_j above). Changing the content selector does not have a large impact on the performance of the model and we chose to keep the same model for all of our results. The full baseline model is therefore similar to the current state-of-the-art bottom-up model from Gehrmann et al. [2018].

The summarization model made available in OpenNMT-py toolkit [1] includes an implementation of a pointer-generator model. Therefore, we chose to conduct the various preliminary experiments using OpenNMT-py, and we then proceeded to implement our improvements to the pointer-generator model by modifying the toolkit code. In retrospect, this was a poor choice, as specific changes and experiments required modification to all the coupled elements of the toolkit, and surely required more work than if they had of been developed from scratch as single-purpose PyTorch modules.

While using the toolkit made it easier to perform some experiments, there was a learning curve to understanding the framework and we lost time trying to implement an end-to-end model, in the end, which we could not manage to debug and abandoned. Also, in the current OpenNMT-py toolkit, the inclusion of a coverage vector is not fully implemented, so we needed to spend time ensuring that the coverage vector was effective passed to the attention calculations. In sum, we spent a significant amount of time digging into the toolkit code.

[1] The OpenNMT-py toolkit is available at `http://opennmt.net/OpenNMT-py/Summarization.html`. The OpenNMT project is based on PyTorch and is described in Klein et al. [2017].

The inputs and outputs for the two-step bottom-up model implementation of Gehrmann et al. [2018] is available from the author's gitHub repo [2]. The AllenNLP toolkit is used to train the content selection tagging model. However the full pipeline of the bottom-up model has not yet been implemented within the OpenNMT-py toolkit, and the code to provide the linkage requires running a legacy fork of the OpenNMT toolkit. Therefore, we implemented the corresponding changes in the current version of OpenNMT-py before continuing with our improvements.

3 Experiments

3.1 Data

All the experiments so far have been conducted with the CNN/Daily Mail dataset that has become a benchmark for the purpose of evaluating neural summarizers [Hermann et al. [2015] and Nallapati et al. [2016]]. It was the main dataset in both Gehrmann et al. [2018] and See et al. [2017]. It is composed of pairs of source online news articles and their summaries with a total of 287,226 training examples, 13,368 validation examples and 11,490 test examples. The source articles contain up to 800 tokens, and 766 tokens on average. The full-sentence summaries are created from bullet points which contain 56 tokens, on average.

The dataset was obtained from the Harvard-NLP GitHub repo [3] and then pre-processed to truncate the articles to a maximum of 400 tokens, and truncate the summaries to a maximum of 100 tokens. This means that summaries typically contain about three sentences. To reduce inference time, we reduced the number of test examples to 10,000 in order to fit into a single shard.

3.2 Evaluation method

Following the majority of papers on abstractive summarization, we evaluate performance using the ROUGE evaluation metrics, ROUGE-1, ROUGE-2 and ROUGE-L. The ROUGE-1 and ROUGE-2 are the unigram and bigram recall, respectively, between the candidate sentence summary and the reference summary. The ROUGE-L is a metric of the longest common subsequence, which scores the in-sequence unigram matches between the candidate summary and the reference summary. As with the BLEU metric, these are automatic, repeatable metrics but they also suffer limitations by not taking into account other aspects of abstraction quality, such as fluency and coherence. To measure the originality of the summary, we also calculate the ratios of novel words, and compare these to extractive model and human summary baselines.

As the objective of abstractive summarization is to generate relevant content, not simply copy-paste, we also look at the share measure of novel content. We consider the share of new single words, bigrams and trigrams generated in the summary which are not part of the source article. We calculate these over the entire set of test examples and report the average share and the maximum share. For extractive summaries this metric will be zero, as all other the words are extracted. By contrast, the set of reference summaries contain 20.27% new words, rising to a maximum of 87.67%. The summaries contain 55.97% new bigrams (maximum 100%) and 73.60% new trigrams (maximum 100%).

3.3 Experimentation details

We used several platforms to conduct the analysis: a GeForce GTX 1080 single-GPU in a local machine for development and tuning. Training the baseline pointer-generator network fully to 200,000 steps requires approximately two full days. For the purposed of conducting experiments, we employed several concurrent AWS GPU instances, over varying performance: NVIDIA Tesla V100 GPUs (P3 instances), K80 GPUs (P2 instances) and M60 GPUs (G3 instances). Interestingly, only the V100 GPUs were faster than the local machine. In order to reduce training times, the models used for the ablation studies were limited to 100,000 steps each. The inference times for generating summaries for evaluation were equally long, roughly 90 minutes on the V100 GPU, and we reduced the test set to 10,000 samples which eliminated the need for multiple data shards.

[2] (`https://github.com/sebastianGehrmann/bottom-up-summary`)

[3] (`https://github.com/harvardnlp/sent-summary`)

Our experiments were conducted changing one aspect at a time, in a reverse ablation study method building on the baseline model. We began using Adam optimization with learning rate 0.001 ($\beta_1 = 0.9$, $\beta_2 = 0.999$), for a single-layer model for the LSTMs and additive attention scores. We then continued by increasing the number of LSTM layers in both the encoder and the decoder from 1 to 3, but keeping the same number of hidden states, in order to see if there were benefits from using a multi-layer seq2seq architecture. Next, we changed the attention score function for both the encoder and decoder from additive to multiplicative. Lastly, we tried dropout regularization with dropout probability $p = 0.2$ (tuned using 2,000 sample test sets). In previous experimentation, we had established that changing (increasing) the dimensions of the word embeddings in the model did not improve performance.

Next, we trained a single-layer model including a coverage vector on the attention mechanism, and reverted to using additive attention (we did not have time to retrain with multiplicative attention). The last experiment was to keep the same model but to change the optimization method from Adam to AdaGrad with learning rate 0.15. For these last two models, we trained on 200,000 steps, in line with the training of the baseline model.

3.4 Results

The results of the experiments show progress in the evaluation metrics of the models as we added or changed architecture choices. This is a sign that with additional experimentation with the models, we could expect further improved scores. The ROUGE scores for the experiments are shown in (Table 1). While there is progress established in the performance of the models relative to each other, none of the models surpasses the Bottom-up model benchmark.

Table 1: Model's ROUGE scores and the Bottom-up benchmark (%)

Model	**R-1**	**R-2**	**R-L**
1-layer – additive attention – Adam	38.810	17.343	36.263
3-layer – additive attention – Adam	39.501	17.897	36.720
3-layer – multiplicative attention – Adam	39.609	17.812	36.775
3-layer – multiplicative attention – 0.2 dropout – Adam	39.939	18.060	37.060
1-layer – additive attention – coverage – Adam	39.975	**18.064**	**37.067**
1-layer – additive attention – coverage – Adagrad	**40.214**	17.965	37.064
1-layer – Bottom-up model (Estimated by paper authors)	**41.277**	**18.937**	**38.391**

While there is progress, none of the pointer-generator experiments individually showed a strong performance boost. Going to a 3-layer bi-LSTM, and adding dropout regularisation, do increase the ROUGE score, but these gains are eclipsed by the inclusion of coverage. Overall, these results were not expected. We expected that one of the experiments would lead to a more significant increase. While we noted that introducing the attention coverage mechanism leads to a small improvement, it does not result in a large change in ROUGE score. In fact, the ROUGE metric does not directly measure repeated words or phrases, so the lack of gain is understandable.

Table 2: Share of new words, bigrams and trigrams in the summary (%)

Model	**words**		**bigrams**		**trigrams**	
	avg	**max**	**avg**	**max**	**avg**	**max**
1-layer – add. attn. – Adam	1.23	47.83	10.32	89.29	19.76	96.67
3-layer – add. attn. – Adam	0.80	43.75	8.09	86.67	16.11	100
3-layer – mult. attn. – Adam	0.51	40.70	7.65	87.80	15.51	97.14
3-layer – mult. attn. – 0.2 dropout – Adam	0.41	46.15	6.13	82.85	12.81	100
1-layer – add. attn.– coverage – Adam	0.59	44.44	6.58	79.41	13.41	96.97
1-layer – add. attn.– coverage – Adagrad	0.08	48.00	2.58	85.29	6.36	100
1-layer – Bottom-up model (Paper authors)	0.94	62.50	10.79	91.89	21.50	100
Reference summary	20.27	87.77	55.97	100	73.60	100

The measures of the share of novel content for single words, bigrams and trigrams generated in the summary are surprising, as they show a very low proportion of new single words (Table 2). As the

maximum length of the summary is 100 words (and many are smaller) the shares reported for all the models, including the Bottom-up baseline model, indicate that on average, only 1 new word is generated per sentence. Even the sentence with the most new words, still only generates about half of the summary from new words. By contrast, on average, one fifth of the reference summary words are not in the source document.

There is a higher share of novel bigrams and trigrams, which shows that the models are not entirely just copy-pasting. We find that, generally, the models with the highest share of novel bigrams and trigrams are those with the lowest ROUGE score. These results just confirm that there is a large gap between the state-of-the-art in neural summarization and summaries produced by humans.

4 Evaluation

The experimentation work made it clear, as is shown also in results tables, that gains in ROUGE score were at the expense of novelty and originality in the summary. On this dataset, the simple LEAD-3 method of summarizing by taking the leading 3 sentences of the article is able to achieve an R-1 score of 40.1. As models were experimented with, higher ROUGE score corresponded to higher copy-attention. The main failure of our models is the generation of new words. Our models remain highly extractive, and the main difference between them is the sets of phrases that are extracted. This is illustrated by looking at the two examples annexed to the paper.

In the first example, the last sentence of the reference summary contains elements that were truncated from the original. And the summary has only two other short sentences. In this case, it would tend to be difficult for any summary to have a high ROUGE score. The 1-layer Bottom-up model and the 1-layer additive attention model both extract more detailed, perhaps superfluous, content in the first two sentences. But the last sentence contains an extracted phrase with an important element of the reference summary *traces of bacteria*.

In fact, as a first sentence, all the summaries have by and large extracted the second sentence of the article. This is due to attention, since this sentence is mostly repeated in the article, and therefore these words will have a high probability in the attention distribution and also a high probability from the content-selection model prediction. Despite the high attention, none of these words are repeated.

Our model's summaries of the first example, while nonetheless being composed of sentences from different parts of the source text, have a high readability. We attribute this to the combined attention on both the encoder and decoder. The final summary is short, but highly readable and in reviewing the original article, provides a perhaps more relevant summary of the article than the given reference summary does. This points to the drawbacks in using a ROUGE score with a single reference, and its lack of fluency and coherence measurement.

The second example is an article about a race horse that wanders off and gets stuck in the mud, but the reference summary is a brief headline type and the source article actually contains repeated headline elements in its last sentences. The Bottom-up model fails to capture that the animal was a horse, despite this being mentioned several times and the reference to 'hoof'. Again none of the neural summaries mention the last phrase of the reference (seen by vet and back to stables). This is perhaps less relevant in the article than the description of the hard working fire crew that rescued the horse. This is the content that the neural models pick up.

All three of our models summarize with two of the same sentences about the fire crew, which we ascribe to increased attention on these elements of the source article. Once the summary has included the first sentence about the crew, the probability of selecting the second sentence about the borough commander receives additional attention from the decoder. However, the Bottom-up model lacks this co-attention and its summary is both less relevant and less coherent. Once again, our final summary is both readable and contains the relevant information from the article. However, it is missing the generative component, and this is one area for future improvement of the models.

5 Conclusion

This work provides two extensions to the attention mechanism of the bottom-up approach, through adding intra-attention on past states of the decoder to the pointer-generator architecture and incorporating the prediction probabilities from the content-selector directly in the "copy-attention"

mechanism. While we do not push the bar of the state-of-the-art in terms of ROUGE score, we find that the resulting summaries have a high readability and are able to extract the phrases of the source article which are inter-related or reflect elements that have been stressed more than once.

The key area for improvement is on the generation of novel words and phrases. While this is unlikely to increase the single-reference ROUGE score, we could envisage a human evaluation metric to assess abstractive performance. In future work, we intend to integrate content selection into an end-to-end model in which the selection objective is jointly optimized with the summarization objective.

6 Additional information - Mentor: Abigail See

7 Examples

Reference summary *eileen dee was being treated for cancer when she caught the lethal bug. her husband rene has spoken of watching her die just five days later. several hospital rooms , including eileen 's , found to have traces of bacteria. mr dee is now suing the hospital trust in brighton for clinical negligence.*
1-layer Bottom-up model with coverage retired chief executive ren dee watched as his wife eileen , 68 , slipped away just days after catching the infection and before the couple 's son had a chance to say goodbye. eileen had been battling cancer when she caught the bug while undergoing treatment at the royal sussex county hospital in brighton. several hospital rooms were found to have traces of the bacteria in the shower drain.
1-layer – additive attention – Adam rene dee watched as his wife eileen , 68 , slipped away just days after catching the infection. the 69-year-old said : ' it was such a shock , one minute everyone was telling her how well she was doing and the next moment i am watching her struggling to breathe ' several hospital rooms, including eileen's , were found to have traces of the drug-resistant bacteria pseudomonas in water running from the tap and in the shower drain.
3-layer – multiplicative attention – dropout – Adam retired chief executive rene dee watched as his wife eileen , 68 , slipped away just days after catching a lethal infection . she had been battling cancer when she caught the bug while undergoing treatment at the royal sussex county hospital in brighton . eileen had been intent on just receiving palliative care because she had watched her mother die of cancer.
1-layer – additive attention – coverage – Adagrad retired chief executive ren dee watched as his wife eileen , 68 , slipped away just days after catching the infection and before the couple 's son had a chance to say goodbye . eileen had received a positive prognosis while being treated for her cancer but died because of an infection at the royal sussex county hospital in brighton.

Reference summary *cody got stuck after wandering into ditch in belvedere , south-east london . animal rescue experts spent an hour and a half rescuing distressed horse. after being freed , the horse was seen by a vet and taken back to his stables*
1-layer Bottom-up model with coverage crew members climbed into the watery ditch as they tried to ease the animal to safety. cody got stuck in the mud after wandering into a bog in belvedere south-east london. cody was covered head to hoof in mud.
1-layer – additive attention – Adam crew members climbed into the watery ditch as they tried to ease the distressed animal to safety on friday afternoon. fire crews were dispatched to save him from his ordeal shortly before 1.30 pm. borough commander for bexley richard welch said crews worked hard to save the horse.
3-layer – multiplicative attention – dropout – Adam cody got stuck in the mud after wandering into a bog in belvedere , south-east london . crew members climbed into the watery ditch as they tried to ease the animal to safety on friday afternoon. borough commander for bexley richard welch said crews worked hard to save the horse .
1-layer – additive attention – coverage – Adagrad cody somehow found himself stuck in the bog in belvedere , south-east london. crew members climbed into the watery ditch as they tried to ease the distressed animal to safety on friday afternoon. borough commander for bexley richard welch said crews worked hard to save the horse.

References

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. Bottom-up abstractive summarization. *CoRR*, abs/1808.10792, 2018. URL http://arxiv.org/abs/1808.10792.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL http://arxiv.org/abs/1506.03340.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. A unified model for extractive and abstractive summarization using inconsistency loss. *CoRR*, abs/1805.06266, 2018. URL http://arxiv.org/abs/1805.06266.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810, 2017. URL http://arxiv.org/abs/1701.02810.

Wojciech Kryscinski, Romain Paulus, Caiming Xiong, and Richard Socher. Improving abstraction in text summarization. *CoRR*, abs/1808.07913, 2018. URL http://arxiv.org/abs/1808.07913.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023, 2016. URL http://arxiv.org/abs/1602.06023.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017. URL http://arxiv.org/abs/1704.04368.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and Mohammad Norouzi et. al. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.