Get To The Point: Summarization with Pointer-Generator Networks

Abigail See
Stanford University
abisee@stanford.edu

Peter J. Liu
Google Brain
peterjliu@google.com

Christopher D. Manning
Stanford University
manning@stanford.edu

Abstract

Neural sequence-to-sequence models have provided a viable new approach for *abstractive* text summarization (meaning they are not restricted to simply selecting and rearranging passages from the original text). However, these models have two shortcomings: they are liable to reproduce factual details inaccurately, and they tend to repeat themselves. In this work we propose a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways. First, we use a hybrid pointer-generator network that can copy words from the source text via *pointing*, which aids accurate reproduction of information, while retaining the ability to produce novel words through the *generator*. Second, we use *coverage* to keep track of what has been summarized, which discourages repetition. We apply our model to the *CNN / Daily Mail* summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points.

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, ***muhammadu buhari*** told cnn's christiane amanpour that **he plans to aggressively fight corruption that has long plagued nigeria** and go after the root of the nation's unrest. *buhari* said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, **he said his administration is confident it will be able to thwart criminals** and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. *buhari* defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. **the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.**

Baseline Seq2Seq + Attention: UNK UNK says his administration is confident it will be able to **destabilize nigeria's economy**. UNK says his administration is confident it will be able to thwart criminals and other nigerians. **he says the country has long nigeria and nigeria's economy.**

Pointer-Gen: *muhammadu buhari* says he plans to aggressively fight corruption **in the northeast part of nigeria**. he says he'll "rapidly give attention" to curbing violence **in the northeast part of nigeria**. he says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Figure 1: Comparison of output of 3 abstractive summarization models on a news article. The baseline model makes **factual errors**, a **nonsensical sentence** and struggles with OOV words *muhammadu buhari*. The pointer-generator model is accurate but **repeats itself**. Coverage eliminates repetition. The final summary is composed from **several fragments**.

1 Introduction

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original. There are two broad approaches to summarization: *extractive* and *abstractive*. *Extractive methods* assemble summaries exclusively from passages (usually whole sentences) taken directly from the source text, while *abstractive methods* may generate novel words and phrases not featured in the source text – as a human-written abstract usually does. The extractive approach is easier, because copying large chunks of text from the source document ensures baseline levels of grammaticality and accuracy. On the other hand, sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework (see Figure 5).

Due to the difficulty of abstractive summarization, the great majority of past work has been extractive (Kupiec et al., 1995; Paice, 1990; Saggion and Poibeau, 2013). However, the recent success of *sequence-to-sequence* models (Sutskever

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1073–1083
Vancouver, Canada, July 30 - August 4, 2017. ©2017 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17-1099

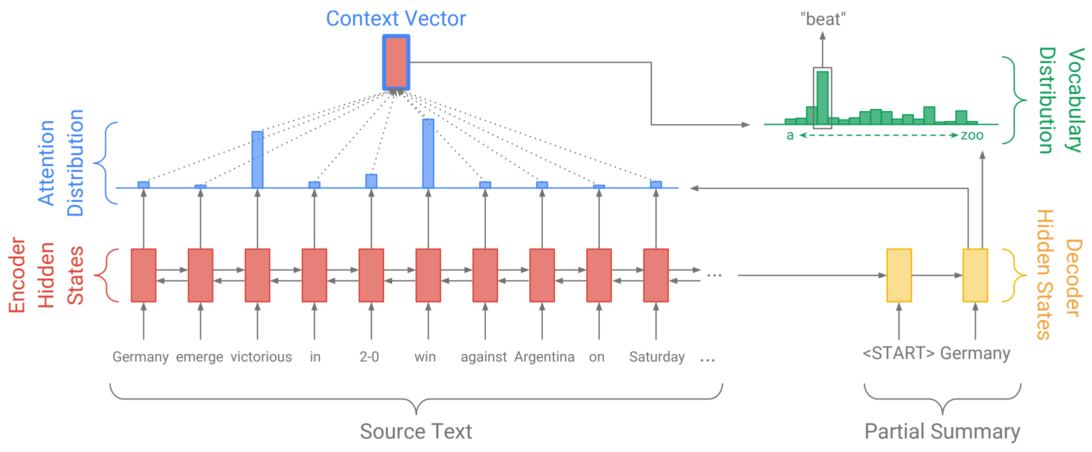

Figure 2: Baseline sequence-to-sequence model with attention. The model may attend to relevant words in the source text to generate novel words, e.g., to produce the novel word *beat* in the abstractive summary *Germany* ***beat*** *Argentina 2-0* the model may attend to the words *victorious* and *win* in the source text.

et al., 2014), in which recurrent neural networks (RNNs) both read and freely generate text, has made abstractive summarization viable (Chopra et al., 2016; Nallapati et al., 2016; Rush et al., 2015; Zeng et al., 2016). Though these systems are promising, they exhibit undesirable behavior such as inaccurately reproducing factual details, an inability to deal with out-of-vocabulary (OOV) words, and repeating themselves (see Figure 1).

In this paper we present an architecture that addresses these three issues in the context of multi-sentence summaries. While most recent abstractive work has focused on headline generation tasks (reducing one or two sentences to a single headline), we believe that longer-text summarization is both more challenging (requiring higher levels of abstraction while avoiding repetition) and ultimately more useful. Therefore we apply our model to the recently-introduced *CNN/ Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains news articles (39 sentences on average) paired with multi-sentence summaries, and show that we outperform the state-of-the-art abstractive system by at least 2 ROUGE points.

Our hybrid *pointer-generator* network facilitates copying words from the source text via *pointing* (Vinyals et al., 2015), which improves accuracy and handling of OOV words, while retaining the ability to *generate* new words. The network, which can be viewed as a balance between extractive and abstractive approaches, is similar to Gu et al.'s (2016) CopyNet and Miao and Blunsom's (2016) Forced-Attention Sentence Compression, that were applied to short-text summarization. We propose a novel variant of the *coverage vector* (Tu et al., 2016) from Neural Machine Translation, which we use to track and control coverage of the source document. We show that coverage is remarkably effective for eliminating repetition.

2 Our Models

In this section we describe (1) our baseline sequence-to-sequence model, (2) our pointer-generator model, and (3) our coverage mechanism that can be added to either of the first two models. The code for our models is available online.[1]

2.1 Sequence-to-sequence attentional model

Our baseline model is similar to that of Nallapati et al. (2016), and is depicted in Figure 2. The tokens of the article w_i are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of *encoder hidden states* h_i. On each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder), and has *decoder state* s_t. The *attention distribution* a^t is calculated as in Bahdanau et al. (2015):

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \quad (1)$$
$$a^t = \text{softmax}(e^t) \quad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters. The attention distribution can be viewed as

[1]www.github.com/abisee/pointer-generator

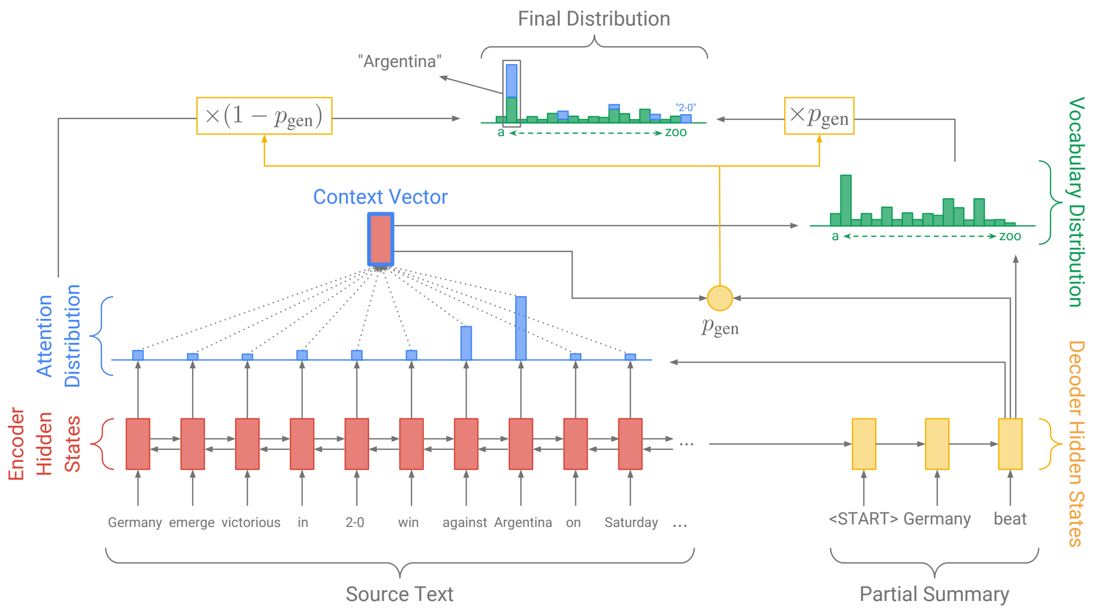

Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0,1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article words such as *2-0* are included in the final distribution. Best viewed in color.

a probability distribution over the source words, that tells the decoder where to look to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the *context vector* h_t^*:

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

The context vector, which can be seen as a fixed-size representation of what has been read from the source for this step, is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}:

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \tag{4}$$

where V, V', b and b' are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and provides us with our final distribution from which to predict words w:

$$P(w) = P_{\text{vocab}}(w) \tag{5}$$

During training, the loss for timestep t is the negative log likelihood of the target word w_t^* for that timestep:

$$\text{loss}_t = -\log P(w_t^*) \tag{6}$$

and the overall loss for the whole sequence is:

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T} \text{loss}_t \tag{7}$$

2.2 Pointer-generator network

Our pointer-generator network is a hybrid between our baseline and a pointer network (Vinyals et al., 2015), as it allows both copying words via pointing, and generating words from a fixed vocabulary. In the pointer-generator model (depicted in Figure 3) the attention distribution a^t and context vector h_t^* are calculated as in section 2.1. In addition, the *generation probability* $p_{\text{gen}} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \tag{8}$$

where vectors w_{h^*}, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Next, p_{gen} is used as a soft switch to choose between *generating* a word from the vocabulary by sampling from P_{vocab}, or *copying* a word from the input sequence by sampling from the attention distribution a^t. For each document let the *extended vocabulary* denote the union of the vocabulary, and all words appearing in the source document. We obtain the following probability distribution over the extended vocabulary:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1-p_{\text{gen}}) \sum_{i:w_i=w} a_i^t \tag{9}$$

Note that if w is an out-of-vocabulary (OOV) word, then $P_{\text{vocab}}(w)$ is zero; similarly if w does not appear in the source document, then $\sum_{i:w_i=w} a_i^t$ is zero. The ability to produce OOV words is one of the primary advantages of pointer-generator models; by contrast models such as our baseline are restricted to their pre-set vocabulary.

The loss function is as described in equations (6) and (7), but with respect to our modified probability distribution $P(w)$ given in equation (9).

2.3 Coverage mechanism

Repetition is a common problem for sequence-to-sequence models (Tu et al., 2016; Mi et al., 2016; Sankaran et al., 2016; Suzuki and Nagata, 2016), and is especially pronounced when generating multi-sentence text (see Figure 1). We adapt the *coverage model* of Tu et al. (2016) to solve the problem. In our coverage model, we maintain a *coverage vector* c^t, which is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{10}$$

Intuitively, c^t is a (unnormalized) distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far. Note that c^0 is a zero vector, because on the first timestep, none of the source document has been covered.

The coverage vector is used as extra input to the attention mechanism, changing equation (1) to:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}) \tag{11}$$

where w_c is a learnable parameter vector of same length as v. This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarized in c^t). This should make it easier for the attention mechanism to avoid repeatedly attending to the same locations, and thus avoid generating repetitive text.

We find it necessary (see section 5) to additionally define a *coverage loss* to penalize repeatedly attending to the same locations:

$$\text{covloss}_t = \sum_i \min(a_i^t, c_i^t) \tag{12}$$

Note that the coverage loss is bounded; in particular $\text{covloss}_t \leq \sum_i a_i^t = 1$. Equation (12) differs from the coverage loss used in Machine Translation. In MT, we assume that there should be a roughly one-to-one translation ratio; accordingly the final coverage vector is penalized if it is more or less than 1. Our loss function is more flexible: because summarization should not require uniform coverage, we only penalize the overlap between each attention distribution and the coverage so far – preventing repeated attention. Finally, the coverage loss, reweighted by some hyperparameter λ, is added to the primary loss function to yield a new composite loss function:

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t) \tag{13}$$

3 Related Work

Neural abstractive summarization. Rush et al. (2015) were the first to apply modern neural networks to abstractive text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets. Their approach, which is centered on the attention mechanism, has been augmented with recurrent decoders (Chopra et al., 2016), Abstract Meaning Representations (Takase et al., 2016), hierarchical networks (Nallapati et al., 2016), variational autoencoders (Miao and Blunsom, 2016), and direct optimization of the performance metric (Ranzato et al., 2016), further improving performance on those datasets.

However, large-scale datasets for summarization of *longer* text are rare. Nallapati et al. (2016) adapted the DeepMind question-answering dataset (Hermann et al., 2015) for summarization, resulting in the *CNN/Daily Mail* dataset, and provided the first abstractive baselines. The same authors then published a neural *extractive* approach (Nallapati et al., 2017), which uses hierarchical RNNs to select sentences, and found that it significantly outperformed their abstractive result with respect to the ROUGE metric. To our knowledge, these are the only two published results on the full dataset.

Prior to modern neural methods, abstractive summarization received less attention than extractive summarization, but Jing (2000) explored cutting unimportant parts of sentences to create summaries, and Cheung and Penn (2014) explore sentence fusion using dependency trees.

Pointer-generator networks. The pointer network (Vinyals et al., 2015) is a sequence-to-sequence model that uses the soft attention distribution of Bahdanau et al. (2015) to produce an output sequence consisting of elements from the input sequence. The pointer network has been used to create hybrid approaches for NMT (Gulcehre et al., 2016), language modeling (Merity et al., 2016), and summarization (Gu et al., 2016; Gulcehre et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016; Zeng et al., 2016).

Our approach is close to the Forced-Attention Sentence Compression model of Miao and Blunsom (2016) and the CopyNet model of Gu et al. (2016), with some small differences: (i) We calculate an explicit switch probability p_{gen}, whereas Gu et al. induce competition through a shared softmax function. (ii) We recycle the attention distribution to serve as the copy distribution, but Gu et al. use two separate distributions. (iii) When a word appears multiple times in the source text, we sum probability mass from all corresponding parts of the attention distribution, whereas Miao and Blunsom do not. Our reasoning is that (i) calculating an explicit p_{gen} usefully enables us to raise or lower the probability of all generated words or all copy words at once, rather than individually, (ii) the two distributions serve such similar purposes that we find our simpler approach suffices, and (iii) we observe that the pointer mechanism often copies a word while attending to multiple occurrences of it in the source text.

Our approach is considerably different from that of Gulcehre et al. (2016) and Nallapati et al. (2016). Those works train their pointer components to activate only for out-of-vocabulary words or named entities (whereas we allow our model to freely learn when to use the pointer), and they do not mix the probabilities from the copy distribution and the vocabulary distribution. We believe the mixture approach described here is better for abstractive summarization – in section 6 we show that the copy mechanism is vital for accurately reproducing rare but in-vocabulary words, and in section 7.2 we observe that the mixture model enables the language model and copy mechanism to work together to perform abstractive copying.

Coverage. Originating from Statistical Machine Translation (Koehn, 2009), coverage was adapted for NMT by Tu et al. (2016) and Mi et al. (2016), who both use a GRU to update the coverage vector each step. We find that a simpler approach – summing the attention distributions to obtain the coverage vector – suffices. In this respect our approach is similar to Xu et al. (2015), who apply a coverage-like method to image captioning, and Chen et al. (2016), who also incorporate a coverage mechanism (which they call 'distraction') as described in equation (11) into neural summarization of longer text.

Temporal attention is a related technique that has been applied to NMT (Sankaran et al., 2016) and summarization (Nallapati et al., 2016). In this approach, each attention distribution is divided by the sum of the previous, which effectively dampens repeated attention. We tried this method but found it too destructive, distorting the signal from the attention mechanism and reducing performance. We hypothesize that an early intervention method such as coverage is preferable to a post hoc method such as temporal attention – it is better to *inform* the attention mechanism to help it make better decisions, than to *override* its decisions altogether. This theory is supported by the large boost that coverage gives our ROUGE scores (see Table 1), compared to the smaller boost given by temporal attention for the same task (Nallapati et al., 2016).

4 Dataset

We use the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains online news articles (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average). We used scripts supplied by Nallapati et al. (2016) to obtain the same version of the the data, which has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. Both the dataset's published results (Nallapati et al., 2016, 2017) use the *anonymized* version of the data, which has been pre-processed to replace each named entity, e.g., *The United Nations*, with its own unique identifier for the example pair, e.g., `@entity5`. By contrast, we operate directly on the original text (or *non-anonymized* version of the data),[2] which we believe is the favorable problem to solve because it requires no pre-processing.

5 Experiments

For all experiments, our model has 256-dimensional hidden states and 128-dimensional word embeddings. For the pointer-generator models, we use a vocabulary of 50k words for both source and target – note that due to the pointer network's ability to handle OOV words, we can use

[2]at `www.github.com/abisee/pointer-generator`

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/syn/para
abstractive model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extractive model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

Table 1: ROUGE F_1 and METEOR scores on the test set. Models and baselines in the top half are abstractive, while those in the bottom half are extractive. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script. The METEOR improvement from the 50k baseline to the pointer-generator model, and from the pointer-generator to the pointer-generator+coverage model, were both found to be statistically significant using an approximate randomization test with $p < 0.01$.

a smaller vocabulary size than Nallapati et al.'s (2016) 150k source and 60k target vocabularies. For the baseline model, we also try a larger vocabulary size of 150k.

Note that the pointer and the coverage mechanism introduce very few additional parameters to the network: for the models with vocabulary size 50k, the baseline model has 21,499,600 parameters, the pointer-generator adds 1153 extra parameters (w_{h^*}, w_s, w_x and b_{ptr} in equation 8), and coverage adds 512 extra parameters (w_c in equation 11).

Unlike Nallapati et al. (2016), we do not pretrain the word embeddings – they are learned from scratch during training. We train using Adagrad (Duchi et al., 2011) with learning rate 0.15 and an initial accumulator value of 0.1. (This was found to work best of Stochastic Gradient Descent, Adadelta, Momentum, Adam and RMSProp). We use gradient clipping with a maximum gradient norm of 2, but do not use any form of regularization. We use loss on the validation set to implement early stopping.

During training and at test time we truncate the article to 400 tokens and limit the length of the summary to 100 tokens for training and 120 tokens at test time.[3] This is done to expedite training and testing, but we also found that truncating the article can *raise* the performance of the model (see section 7.1 for more details). For training, we found it efficient to start with highly-truncated sequences, then raise the maximum length once converged. We train on a single Tesla K40m GPU with a batch size of 16. At test time our summaries are produced using beam search with beam size 4.

We trained both our baseline models for about 600,000 iterations (33 epochs) – this is similar to the 35 epochs required by Nallapati et al.'s (2016) best model. Training took 4 days and 14 hours for the 50k vocabulary model, and 8 days 21 hours for the 150k vocabulary model. We found the pointer-generator model quicker to train, requiring less than 230,000 training iterations (12.8 epochs); a total of 3 days and 4 hours. In particular, the pointer-generator model makes much quicker progress in the early phases of training. To obtain our final coverage model, we added the coverage mechanism with coverage loss weighted to $\lambda = 1$ (as described in equation 13), and trained for a further 3000 iterations (about 2 hours). In this time the coverage loss converged to about 0.2, down from an initial value of about 0.5. We also tried a more aggressive value of $\lambda = 2$; this reduced coverage loss but increased the primary loss function, thus we did not use it.

We tried training the coverage model without the loss function, hoping that the attention mechanism may learn by itself not to attend repeatedly to the same locations, but we found this to be ineffective, with no discernible reduction in repetition. We also tried training with coverage from the first

[3]The upper limit of 120 is mostly invisible: the beam search algorithm is self-stopping and almost never reaches the 120th step.

iteration rather than as a separate training phase, but found that in the early phase of training, the coverage objective interfered with the main objective, reducing overall performance.

6 Results

6.1 Preliminaries

Our results are given in Table 1. We evaluate our models with the standard ROUGE metric (Lin, 2004b), reporting the F_1 scores for ROUGE-1, ROUGE-2 and ROUGE-L (which respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated). We obtain our ROUGE scores using the `pyrouge` package.[4] We also evaluate with the METEOR metric (Denkowski and Lavie, 2014), both in exact match mode (rewarding only exact matches between words) and full mode (which additionally rewards matching stems, synonyms and paraphrases).[5]

In addition to our own models, we also report the lead-3 baseline (which uses the first three sentences of the article as a summary), and compare to the only existing abstractive (Nallapati et al., 2016) and extractive (Nallapati et al., 2017) models on the full dataset. The output of our models is available online.[6]

Given that we generate plain-text summaries but Nallapati et al. (2016; 2017) generate anonymized summaries (see Section 4), our ROUGE scores are not strictly comparable. There is evidence to suggest that the original-text dataset may result in higher ROUGE scores in general than the anonymized dataset – the lead-3 baseline is higher on the former than the latter. One possible explanation is that multi-word named entities lead to a higher rate of *n*-gram overlap. Unfortunately, ROUGE is the only available means of comparison with Nallapati et al.'s work. Nevertheless, given that the disparity in the lead-3 scores is (+1.1 ROUGE-1, +2.0 ROUGE-2, +1.1 ROUGE-L) points respectively, and our best model scores exceed Nallapati et al. (2016) by (+4.07 ROUGE-1, +3.98 ROUGE-2, +3.73 ROUGE-L) points, we may estimate that we outperform the only previous abstractive system by at least 2 ROUGE points all-round.

[4]pypi.python.org/pypi/pyrouge/0.1.3
[5]www.cs.cmu.edu/~alavie/METEOR
[6]www.github.com/abisee/pointer-generator

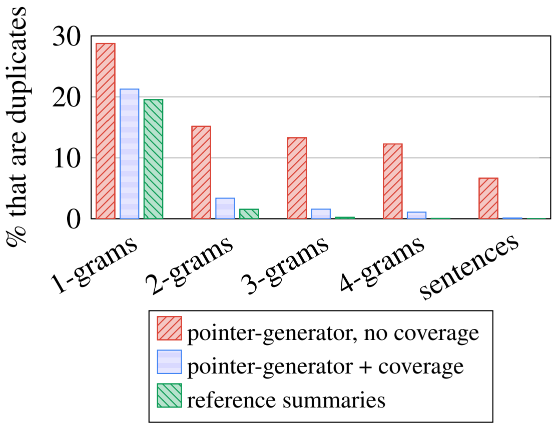

Figure 4: Coverage eliminates undesirable repetition. Summaries from our **non-coverage model** contain many duplicated *n*-grams while our **coverage model** produces a similar number as the **reference summaries**.

6.2 Observations

We find that both our baseline models perform poorly with respect to ROUGE and METEOR, and in fact the larger vocabulary size (150k) does not seem to help. Even the better-performing baseline (with 50k vocabulary) produces summaries with several common problems. Factual details are frequently reproduced incorrectly, often replacing an uncommon (but in-vocabulary) word with a more-common alternative. For example in Figure 1, the baseline model appears to struggle with the rare word *thwart*, producing *destabilize* instead, which leads to the fabricated phrase *destabilize nigeria's economy*. Even more catastrophically, the summaries sometimes devolve into repetitive nonsense, such as the third sentence produced by the baseline model in Figure 1. In addition, the baseline model can't reproduce out-of-vocabulary words (such as *muhammadu buhari* in Figure 1). Further examples of all these problems are provided in the supplementary material.

Our pointer-generator model achieves much better ROUGE and METEOR scores than the baseline, despite many fewer training epochs. The difference in the summaries is also marked: out-of-vocabulary words are handled easily, factual details are almost always copied correctly, and there are no fabrications (see Figure 1). However, repetition is still very common.

Our pointer-generator model with coverage improves the ROUGE and METEOR scores further, convincingly surpassing the best abstractive model

Article: smugglers lure arab and african migrants by offering discounts to get onto overcrowded ships if people bring more potential passengers, a cnn investigation has revealed. (...)
Summary: cnn investigation **uncovers** the **business inside** a **human smuggling ring**.

Article: eyewitness video showing white north charleston police officer michael slager shooting to death an unarmed black man has exposed discrepancies in the reports of the first officers on the scene. (...)
Summary: more **questions than answers emerge** in **controversial s.c.** police shooting.

Figure 5: Examples of highly abstractive reference summaries (**bold** denotes novel words).

of Nallapati et al. (2016) by several ROUGE points. Despite the brevity of the coverage training phase (about 1% of the total training time), the repetition problem is almost completely eliminated, which can be seen both qualitatively (Figure 1) and quantitatively (Figure 4). However, our best model does not quite surpass the ROUGE scores of the lead-3 baseline, nor the current best extractive model (Nallapati et al., 2017). We discuss this issue in section 7.1.

7 Discussion

7.1 Comparison with extractive systems

It is clear from Table 1 that extractive systems tend to achieve higher ROUGE scores than abstractive, and that the extractive lead-3 baseline is extremely strong (even the best extractive system beats it by only a small margin). We offer two possible explanations for these observations.

Firstly, news articles tend to be structured with the most important information at the start; this partially explains the strength of the lead-3 baseline. Indeed, we found that using only the first 400 tokens (about 20 sentences) of the article yielded significantly higher ROUGE scores than using the first 800 tokens.

Secondly, the nature of the task and the ROUGE metric make extractive approaches and the lead-3 baseline difficult to beat. The choice of content for the reference summaries is quite subjective – sometimes the sentences form a self-contained summary; other times they simply showcase a few interesting details from the article. Given that the articles contain 39 sentences on average, there are many equally valid ways to choose 3 or 4 highlights in this style. Abstraction introduces even more options (choice of phrasing), further decreasing the likelihood of matching the reference summary. For example, *smugglers profit from desperate migrants* is a valid alternative abstractive summary for the first example in Figure 5, but it scores 0 ROUGE with respect to the reference summary. This inflexibility of ROUGE is exacerbated by only having one reference summary, which has been shown to lower ROUGE's reliability compared to multiple reference summaries (Lin, 2004a).

Due to the subjectivity of the task and thus the diversity of valid summaries, it seems that ROUGE rewards safe strategies such as selecting the first-appearing content, or preserving original phrasing. While the reference summaries *do* sometimes deviate from these techniques, those deviations are unpredictable enough that the safer strategy obtains higher ROUGE scores on average. This may explain why extractive systems tend to obtain higher ROUGE scores than abstractive, and even extractive systems do not significantly exceed the lead-3 baseline.

To explore this issue further, we evaluated our systems with the METEOR metric, which rewards not only exact word matches, but also matching stems, synonyms and paraphrases (from a predefined list). We observe that all our models receive over 1 METEOR point boost by the inclusion of stem, synonym and paraphrase matching, indicating that they may be performing some abstraction. However, we again observe that the lead-3 baseline is not surpassed by our models. It may be that news article style makes the lead-3 baseline very strong with respect to any metric. We believe that investigating this issue further is an important direction for future work.

7.2 How abstractive is our model?

We have shown that our pointer mechanism makes our abstractive system more reliable, copying factual details correctly more often. But does the ease of copying make our system any less *abstractive*?

Figure 6 shows that our final model's summaries contain a much lower rate of novel n-grams (i.e., those that don't appear in the article) than the reference summaries, indicating a lower degree of abstraction. Note that the baseline model produces novel n-grams more frequently – however, this statistic includes all the incorrectly copied words, *UNK* tokens and fabrications alongside the good instances of abstraction.

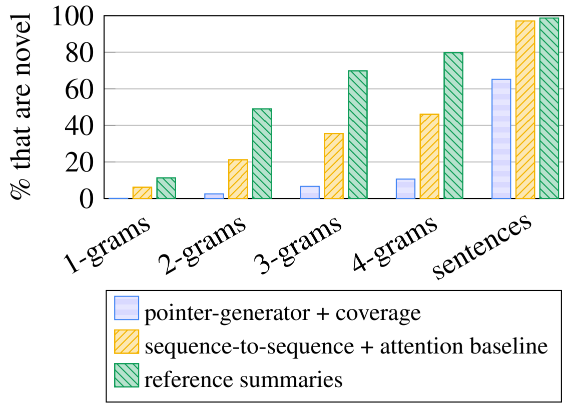

Figure 6: Although our **best model** is abstractive, it does not produce novel n-grams (i.e., n-grams that don't appear in the source text) as often as the **reference summaries**. The **baseline model** produces more novel n-grams, but many of these are erroneous (see section 7.2).

Article: andy murray (...) is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic thiem, who pushed him to 4-4 in the second set before going down 3-6 6-4, 6-1 in an hour and three quarters. (...)
Summary: andy murray **defeated** dominic thiem 3-6 6-4, 6-1 in an hour and three quarters.

Article: (...) wayne rooney smashes home during manchester united 's 3-1 win over aston villa on saturday. (...)
Summary: manchester united **beat** aston villa 3-1 at old trafford on saturday.

Figure 7: Examples of abstractive summaries produced by our model (**bold** denotes novel words).

In particular, Figure 6 shows that our final model copies whole article sentences 35% of the time; by comparison the reference summaries do so only 1.3% of the time. This is a main area for improvement, as we would like our model to move beyond simple sentence extraction. However, we observe that the other 65% encompasses a range of abstractive techniques. Article sentences are truncated to form grammatically-correct shorter versions, and new sentences are composed by stitching together fragments. Unnecessary interjections, clauses and parenthesized phrases are sometimes omitted from copied passages. Some of these abilities are demonstrated in Figure 1, and the supplementary material contains more examples.

Figure 7 shows two examples of more impressive abstraction – both with similar structure. The dataset contains many sports stories whose summaries follow the *X beat Y* ⟨*score*⟩ *on* ⟨*day*⟩ template, which may explain why our model is most confidently abstractive on these examples. In general however, our model does not routinely produce summaries like those in Figure 7, and is not close to producing summaries like in Figure 5.

The value of the generation probability p_{gen} also gives a measure of the abstractiveness of our model. During training, p_{gen} starts with a value of about 0.30 then increases, converging to about 0.53 by the end of training. This indicates that the model first learns to mostly copy, then learns to generate about half the time. However at test time, p_{gen} is heavily skewed towards copying, with a mean value of 0.17. The disparity is likely due to the fact that during training, the model receives word-by-word supervision in the form of the reference summary, but at test time it does not. Nonetheless, the generator module is useful even when the model is copying. We find that p_{gen} is highest at times of uncertainty such as the beginning of sentences, the join between stitched-together fragments, and when producing periods that truncate a copied sentence. Our mixture model allows the network to copy while simultaneously consulting the language model – enabling operations like stitching and truncation to be performed with grammaticality. In any case, encouraging the pointer-generator model to write more abstractively, while retaining the accuracy advantages of the pointer module, is an exciting direction for future work.

8 Conclusion

In this work we presented a hybrid pointer-generator architecture with coverage, and showed that it reduces inaccuracies and repetition. We applied our model to a new and challenging long-text dataset, and significantly outperformed the abstractive state-of-the-art result. Our model exhibits many abstractive abilities, but attaining higher levels of abstraction remains an open research question.

9 Acknowledgment

We thank the ACL reviewers for their helpful comments. This work was begun while the first author was an intern at Google Brain and continued at Stanford. Stanford University gratefully acknowledges the support of the DARPA DEFT Program AFRL contract no. FA8750-13-2-0040. Any opinions in this material are those of the authors alone.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *International Joint Conference on Artificial Intelligence*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Empirical Methods in Natural Language Processing*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *North American Chapter of the Association for Computational Linguistics*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL 2014 Workshop on Statistical Machine Translation*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Association for Computational Linguistics*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Neural Information Processing Systems*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Applied natural language processing*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *International ACM SIGIR conference on Research and development in information retrieval*.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough? In *NACSIS/NII Test Collection for Information Retrieval (NTCIR) Workshop*.

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: ACL workshop*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *NIPS 2016 Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Empirical Methods in Natural Language Processing*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Association for the Advancement of Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çaglar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Computational Natural Language Learning*.

Chris D Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management* 26(1):171–186.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing*.

Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*, Springer, pages 3–21.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Jun Suzuki and Masaaki Nagata. 2016. RNN-based encoder-decoder approach with word frequency estimation. *arXiv preprint arXiv:1701.00138* .

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Empirical Methods in Natural Language Processing*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Association for Computational Linguistics*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Neural Information Processing Systems*.

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. In *International Conference on Machine Learning*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .