# DS-GA 1011 Natural Language Processing with Representation Learning Final Report

*Sam(Huanze) Tang*[1], *Harry(Yaozhong) Huang*[1], *Tanaya Joshi*[1]

[1]New York University

ht2413@nyu.edu, yh2563@nyu.edu, tj2181@nyu.edu

## 1. Introduction

This project is aimed at replicating the existing paper *Get To The Point: Summarization with Pointer-Generator Network*[1]. This paper focused on text summarization, which uses models to generate a short and concise summary that captures the main idea of the source text. Generally, there are two approaches for summarization: extractive and abstractive summarization. Extractive methods create summaries purely from words and sentences taken directly from the source data[2], while abstractive methods may produce new words and phrases not found in the source text like human-written summaries. Compared to the extractive methods, abstractive summarization exhibits more sophisticated abilities that are essential to high-level summarization.

However, due to the complexity of abstractive summarization, abstractive models exhibit many undesirable behaviors: inaccurate reproduction of factual information, inability to deal with out-of-vocabulary words, and generation of repetitive content. The main contribution of the paper is that it proposed an innovative method for solving these three problems faced by abstractive text summarization.

This paper used the *CNN/Daily Mail* dataset which includes news articles paired with multi-sentence summaries[3]. To accurately reproduce the factual details, it proposed a hybrid pointer-generator network that can copy words from the source text via pointing while retaining the ability to produce new phrases and sentences through the generator[4]. To prevent repetition, this paper applied coverage to keep track of the summarized context[5]. More details about the models will be discussed in the following section.

Besides replication of the original paper, we also conducted several experiments on the basis of the point-generator network with a coverage mechanism. We will discuss these experiments in the fourth section.

## 2. Model Description

The model we are using mainly consists of three parts. The first part is the baseline sequence-to-sequence model with attention. The second part is the pointer-generator model and the final part is adding a coverage mechanism to the whole network.

### 2.1. Baseline Seq2Seq with attention

The baseline Seq2Seq model contains an encoder and a decoder. For the encoding part, the tokens of the article $w_i$ are passed into the one single layer BiLSTM encoder one by one[6], and a sequence of *encoder hidden states* $h_i$ will be produced. On each time step of the decoding process, a single layer of the unidirectional LSTM decoder receives the word embedding of the previous word and *decoder hidden state* $s_t$. Moreover, teacher forcing is applied in the training process, which means that instead of using the previous word generated by the decoder, the model uses the ground truth from a prior time step as input during the training. We also implemented additive attention and the attention distribution $a^t$ is calculated as:

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \tag{1}$$

$$a^t = softmax(e^t) \tag{2}$$

The attention distribution can be thought of as a probability distribution over the source words[7], which tells the decoder where to look to generate the next word. With the attention distribution, we further generate a *context vector* $h_i^*$ by calculating a weighted sum of the encoder's hidden state:

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

The context vector could be viewed as a fixed-size representation of what has been read from the source text. It will be concatenated with the decoder hidden state $s_t$ and pass through two linear layers to produce a vocabulary distribution $P_{vocab}$, which represents the probability distribution over all words in the vocabulary:

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

### 2.2. Pointer-generator network

The pointer-generator network adds a pointer network on the basis of our baseline model. It, therefore, allows the model to switch between copying words from the input text and generating a word from the vocabulary[4]. Specifically, the attention distribution $a^t$ and context vector $h_t^*$ are calculated the same way as that in Section 2.1. The main difference is that in this pointer-generator model, a new concept *generation probability* $p_{gen}$ is introduced. $p_{gen}$ helps the model to decide whether *copying* a word from the input source text by sampling from the attention distribution $a^t$ or *generating* a word from the vocabulary by sampling from $P_{vocab}$. At each time $t$, it is calculated using the context vector $h_t^*$, the decoder hidden state $s_t$ and the decoder input $x_t$:

$$p_{gen} = \sigma(W_{h^*}^T h_t^* + W_s^T s_t + W_x^T x_t + b_{ptr}) \tag{5}$$

Let the extended vocabulary represent the union of the vocabulary and all terms found in the source text. The probability distributions of the word $w$ in extended vocabulary can be expressed as:

$$P(w) = p_{gen} P_{vocab} + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{6}$$

According to the equation, if $w$ is an out-of-vocabulary (OOV) word, then $P_{vocab}$ is zero. Similarly, if $w$ does not exist

in the source document, then $\sum_{i:w_i=w} a_i^t$ will be zero. This feature enables the pointer-generator model to produce OOV words and, therefore, outperform the baseline Seq2Seq model that is restricted to the pre-set vocabulary.

### 2.3. Coverage Mechanism

Coverage mechanism prevents the model from generating repetitive summaries[5]. To implement the coverage mechanism, we introduced a *coverage vector* $c^t$, which is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t'=1}^{t-1} a^{t'} \tag{7}$$

Intuitively, $c^t$ is a distribution over the source document words and it represents the degree of coverage that those words have received from the attention mechanism up to this point. Since none of the source documents has been covered at the beginning, $c^0$ should be a zero vector.

The coverage vector is used as an extra input to the model's attention mechanism. As a result, we need change the equation (1) to the following equation:

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + W_c c_i^t + b_{attn}) \tag{8}$$

This will ensure that the attention mechanism is informed by its previous decisions summarized in $c^t$ when making the decision to choose where to attend next. The attention mechanism will consequently be able to avoid repeatedly attending to the locations that have been looked at before. That is the main reason why the coverage mechanism can avoid generating repetitive text.

According to the original paper, *coverage loss* has to be added to penalize repeatedly referring to the same location, or otherwise, it will not address the issue of repetition:

$$covloss_t = \sum_i min(a_i^t, c_i^t) \tag{9}$$

The coverage loss will penalize the overlap between each attention distribution and the coverage so far and hence prevent repeated attention.

Finally, the loss function consists of two parts. The first part is the negative log-likelihood of the timestep $t$'s target word $w_t^*$:

$$loss_{target} = -logP(w_t^*) \tag{10}$$

The second part is adding reweighted coverage loss to the primary loss function $loss_{target}$ to yield a complete loss function:

$$loss_t = -logP(w_t^*) + \lambda \sum_i min(a_i^t, c_i^t) \tag{11}$$

## 3. Dataset

For replication work, we use the *CNN/Daily Mail* dataset [3] imported from the Hugging Face, which is identical to the dataset used in the original paper. The dataset contains 287,226 training pairs, 13,368 validation pairs, and 11,490 test pairs. Each pair consists of one article (781 tokens on average) with a multi-sentence summary (56 tokens on average). After importing the dataset, we tokenized all the sentences without further preprocessing according to the original paper.

Besides replicating the result using the *CNN/Daily Mail* dataset as the original paper does, we applied the model to another Kaggle dataset, which is also a news summary dataset[8]. This dataset is preprocessed by Kondalarao Vonteru, which contains about 98,000 news and corresponding summaries. The difference between the Kaggle dataset and *CNN/Daily Mail* dataset is that the length of the source text in the Kaggle dataset is shorter. We want to check if the model's performance will improve when shortening the source text. Moreover, we also adjusted the RNN structure from LSTM to GRU and applied the GRU structure model to this dataset because GRU usually performs better on short text compared with LSTM.

## 4. Experiments & Results

We implemented our model in PyTorch from scratch by following the mindset of the original paper. We built the Vocabulary class, baseline Seq2Seq model, pointer-generator network without coverage mechanism, and pointer-generator network with coverage mechanism. For the RNN structure in the encoder and decoder, we tested both LSTM and GRU. Moreover, we also implemented *Beam Search* for decoding. In beam search, instead of choosing the best token to generate at each timestep, we keep $k$ possible tokens at each step, which helps our model generate better results.

### 4.1. Replication

We strictly followed the steps in the original papers for the replication and used exactly the same hyperparameters. The original paper did experiments on *CNN/Daily Mail* dataset using three models: the baseline Seq2Seq model with an attention mechanism, the pointer-generator network without a coverage mechanism, and the pointer-generator network with a coverage mechanism. The encoder and decoder structures for these three models are all LSTM. For all the experiments, models have 256-dimensional hidden states and 128-dimensional word embeddings. The pointer-generator models use a vocabulary of 50k words for both source and target, while the baseline model uses a larger vocabulary size of 150k.

During training and testing, all the source articles are truncated to 400 tokens. Meanwhile, the length of the summary is limited to 100 tokens for training and 120 tokens for testing. All the models are trained for 33 epochs with a batch size of 16. The optimizer uses the Adagrad algorithm with a learning rate of 0.15 and an initial accumulator value of 0.1. During the testing, all the summaries are generated using beam search with beam size 4.

Table 1 shows our replication results of baseline Seq2Seq model:

| Metric \ Model | Our Model | Paper's Model |
|---|---|---|
| Training Loss | 5.17 | Unknown |
| ROUGE-1 | 29.46 | 31.33 |
| ROUGE-2 | 10.32 | 11.81 |
| ROUGE-L | 26.71 | 28.83 |

Table 1: *Baseline Seq2Seq*

Table 2 shows our replication results of the pointer-generator network without coverage mechanism:

| Metric \ Model | Our Model | Paper's Model |
|---|---|---|
| Training Loss | 4.63 | Unknown |
| ROUGE-1 | 35.83 | 36.44 |
| ROUGE-2 | 13.94 | 15.66 |
| ROUGE-L | 30.15 | 33.42 |

Table 2: *PGN without coverage mechanism*

Table 3 shows our replication results of pointer-generator network with coverage mechanism:

| Metric \ Model | Our Model | Paper's Model |
|---|---|---|
| Training Loss | 3.28 | Unknown |
| ROUGE-1 | 36.31 | 39.53 |
| ROUGE-2 | 16.04 | 17.28 |
| ROUGE-L | 31.89 | 36.38 |

Table 3: *PGN with LSTM and coverage mechanism*

Although we did not achieve exactly the same results as the paper did, our results are close to the paper's results.

To present our replication results in a straightforward manner, we chose a news article in *CNN/Daily Mail* dataset as an example and generated a summary of this article using our pointer-generator network (PGN) with coverage mechanism model. The truncated original text, reference summary, and model-generated summary are shown below:

**One Example of CNN/Daily Mail Dataset:**

**Original Text (truncated):**
martin o'neill says this summer 's friendly match with england is simply a means to an end for republic of ireland . roy hodgson takes the three lions to dublin on june 7 , six days before the irish face scotland in their decisive euro 2016 qualifier . o'neill , though , has played down the significance of england 's visit and prefers to focus his attention on the match with the scots . martin o'neill says that ireland 's game against scotland is much more important than the england clash . ' the scotland game is the be all and end all . ' he said . i do n't think we should overlook the fact that , for me , it 's a build up to the scotland game . it will be intense and that might be a good thing . o'neill and his team can not afford anything other than victory against scotland at the aviva stadium . failure to win poland, however , has left them three points adrift of the table-topping poles with germany and scotland also two points better off . gordon strachan and his scotland side are just above ireland and the clash has been labelled as ' must win ' the team have been praised for refusing to accept defeat against poland , but also criticised for again having to rely on a last-minute rescue act . o'neill said : ' overall i think the draw sets it up for scotland . it 's probably a must-win game for us .

**Reference Summary:**
martin o'neill says the clash with scotland is bigger than england . ireland and scotland are vying for qualification for euro 2016 in france . o'neill admits that the crucial game with their rivals is ' a must win ' .

**Our Generated Summary:**
roy hodgson takes the three lions to dublin on june 7 , six days . martin o'neill says that ireland 's game against scotland is much more important than the england clash .

### 4.2. Extension

Our extension work mainly consists of three parts. The first part is applying the model to another news dataset. The second part is changing the original encoder and decoder structure from LSTM to another RNN structure, GRU. The third part is using multiplicative attention instead of additive attention in the attention mechanism.

#### *4.2.1. Another new dataset*

The dataset we chose is a news dataset from Kaggle, as introduced in section 3. We applied the same pointer-generator network (PGN) with coverage mechanism to it. The results are as below:

| Metric \ Dataset | Kaggle News |
|---|---|
| Training loss | 2.96 |
| ROUGE-1 | 37.87 |
| ROUGE-2 | 16.64 |
| ROUGE-L | 32.49 |

Table 4: *Experiment on Kaggle News using LSTM structure*

The ROUGE scores are higher than what we got in *CNN/Daily Mail* dataset, which suggests that our model performs better on datasets with shorter source texts. Take the following news as an example, we can tell that the generated summary is very close to the reference summary:

**One Example of Kaggle News Dataset:**

**Original Text (truncated):**
hundreds muslims prayed outside white house friday protest us president donald trumps decision recognise jerusalem israels capital trump piece soil jerusalem palestine owns trump tower give away israelis protesters said notably israel palestine claim jerusalem capital .

**Reference Summary:**
muslims pray outside white house protest jerusalem .

**Our Generated Summary:**
hundreds muslims prayed outside white house recognise jerusalem palestine govt president protest us prez decision .

#### *4.2.2. Changing LSTM to GRU*

We first changed the RNN structure from LSTM to GRU and trained on *CNN/Daily Mail* dataset. The results are as below:

| Metric \ Model | Our Model | Paper's Model |
|---|---|---|
| Training Loss | 3.57 | Unknown |
| ROUGE-1 | 36.11 | 39.53 |
| ROUGE-2 | 15.79 | 17.28 |
| ROUGE-L | 31.53 | 36.38 |

Table 5: *PGN with GRU and coverage mechanism*

According to the table above, turning LSTM into GRU does not improve the performance on *CNN/Daily Mail* dataset. One possible reason is that the articles in *CNN/Daily Mail* dataset are generally long and LSTM works better on long sequences than GRU[9].

Meanwhile, we applied this GRU structure to the Kaggle dataset, which contains shorter source text and the results are as below:

| Metric \ Dataset | Kaggle News |
|---|---|
| Training loss | 2.44 |
| ROUGE-1 | 39.05 |
| ROUGE-2 | 16.92 |
| ROUGE-L | 34.18 |

Table 6: *Experiment on Kaggle News using GRU structure*

The results show that the GRU structure outperforms LSTM on the Kaggle dataset. This aligns with our intuition since GRU tends to have better performance on shorter sequences.

#### *4.2.3. Multiplicative attention*

In addition to additive attention described in equation (8), we also implemented the multiplicative attention. The multiplicative attention in the Seq2Seq model is calculated as below[10]:

$$f_{attn}(h_i, s_j) = h_i^T W_a s_j \tag{12}$$

However, since our model included an additional coverage vector $c^t$, we need to adjust the formula for calculating multiplicative attention as below:

$$e_i^t = h_i^T W_1 s_t + h_i^T W_2 c_i^t + s_t^T W_3 c_i^t + b_{attn} \tag{13}$$

We replace additive attention with multiplicative attention in our PGN model with LSTM and coverage mechanism. The results are shown below:

| Metric \ Attention | Additive | Multiplicative |
|---|---|---|
| Training Loss | 3.28 | 3.32 |
| ROUGE-1 | 36.31 | 36.27 |
| ROUGE-2 | 16.04 | 15.96 |
| ROUGE-L | 31.89 | 31.92 |
| Training time | 52 min/epoch | 44 min/epoch |

Table 7: *Additive vs. Multiplicative*

The results suggest that the multiplicative attention mechanism does not improve the results but it accelerates the training process, saving about 8 minutes for each epoch. Since our training process contains 33 epochs, it saves us around 4 hours for training at the expense of slightly poorer performance.

## 5. Discussion

### 5.1. Analysis of Our Replication & Extension Work

For the replication part, we achieved a result that is close to the results of the original paper but slightly lower. One of the reasons can be that some details in the structure of the model implemented in PyTorch differ from the model built in the original paper using the old version of TensorFlow. Another possible reason is that the initialization of the parameters in PyTorch is different than that in TensorFlow.

As for the extension part, our pointer generator network with LSTM and coverage mechanism performs better on the news dataset from Kaggle that contains shorter source texts. This suggests that our model fits better in short news summarization.

Switching from the LSTM to the GRU structure, we received lower rouge scores. Our explanation of this decrease is because *CNN/Daily Mail* dataset contains long source texts. Generally, LSTM is more appropriate for long text while GRU is born for short sequences. Therefore, it is reasonable that the rouge scores decreased when we applied GRU on *CNN/Daily Mail* dataset while they increased on the Kaggle dataset with short texts.

Finally, we applied multiplicative attention instead of additive attention. Although multiplicative attention does not achieve better results, it turns out to save us about 8 minutes per epoch and over 4 hours for each training process. That is to say, multiplicative attention provides us with a more efficient method of attention mechanism at the expense of slightly poorer performance.

### 5.2. Limitations

Despite the achievements we make, there are still some limitations of the whole model.

First, our model receives relatively low ROUGE scores (only around 20 ROUGE-1 score) on the dialogue dataset. We think this is related to the decoding strategies. Usually, we use *beam search* to explore $k$ high-probability sequences simultaneously. However, there are issues of dullness in dialogue when using classic decoding strategies because the very highest probability outcomes could lead to degenerate text[11]. To solve this problem, we could try other decoding strategies such as nucleus sampling[12].

Second, the model has the limitation that the coverage attention could not learn by itself to reduce the repetition problem. To manually ensure that repetition could be decreased during the training process, the authors of the original paper added a coverage loss into the loss function. However, since the coverage vector is included in the attention mechanism, we expect it to learn to reduce the repetition by itself but it turns out to be ineffective. The reason behind it may require future research.

## 6. Conclusion

In this project, we first replicated the paper *Get To The Point: Summarization with Pointer-Generator Network*[1] and achieved results that are close to the paper's results. Based on the replication, we further conducted several experiments on the dataset, attention mechanism, and RNN structure. The main conclusions could be summarized as follows: *(1)* our model performs better on short texts summarization tasks than on high-level abstraction tasks (e.g. long text and dialogue summarization). *(2)* Models with multiplicative attention can achieve similar performance as additive attention while reducing training time to some extent. *(3)* Models using GRU as the RNN structure performs worse on the original long news dataset but generates better summaries on the short news dataset.

## 7. Author Contribution Statement

We contributed to the project equally, and our work distribution is as below:

| Name | Work |
|---|---|
| Sam Tang | Implement models, experiment on LSTM and GRU, and write reports |
| Harry Huang | Implement models, experiment on new dataset, and write reports |
| Tanaya Joshi | Propose model structure, implement models, and experiment on attention mechanism |

Table 8: *Contribution Statement*

# 8. References


[1] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *CoRR*, vol. abs/1704.04368, 2017. [Online]. Available: http://arxiv.org/abs/1704.04368

[2] M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu, and X. Huang, "Extractive summarization as text matching," *CoRR*, vol. abs/2004.08795, 2020. [Online]. Available: https://arxiv.org/abs/2004.08795

[3] B. Z. Ramesh Nallapati and C. dos Santos, "Dataset: Cnn/daily mail," https://paperswithcode.com/paper/abstractive-text-summarization-using-sequence.

[4] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[5] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Coverage-based neural machine translation," *CoRR*, vol. abs/1601.04811, 2016. [Online]. Available: http://arxiv.org/abs/1601.04811

[6] Ç. Gülçehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," *CoRR*, vol. abs/1603.08148, 2016. [Online]. Available: http://arxiv.org/abs/1603.08148

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] K. Vonteru, "Dataset: Kaggle news," https://www.kaggle.com/code/edumunozsala/pointer-generator-in-pytorch/data.

[9] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[10] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[11] Z. Li, J. Kiseleva, and M. de Rijke, "Improving response quality with backward reasoning in open-domain dialogue systems," *CoRR*, vol. abs/2105.00079, 2021. [Online]. Available: https://arxiv.org/abs/2105.00079

[12] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *CoRR*, vol. abs/1904.09751, 2019. [Online]. Available: http://arxiv.org/abs/1904.09751