An Examination of the CNN/DailyMail Neural Summarization Task

Vincent Chen
Stanford University
vschen@stanford.edu

Eduardo Torres Montaño
Stanford University
torresm9@stanford.edu

Liezl Puzon
Stanford University
puzon@stanford.edu

Advisor: Danqi Chen
Stanford University
danqi@cs.stanford.edu

Abstract

Abstractive word summarization has proven to be difficult as the task involves a good understanding of a language model to reproduce both structurally correct and meaningful sentences. Given these difficult tasks we focus on improving the decoder, in the encoder-decoder neural summarization model. Here we look into several approaches in order to improve an already existing sequence-to-sequence model. Two problems we identified in the existing models are a lack of generalization in using longer articles, and second an inability to use the source text itself to provide words and relying on using <UNK> tokens instead. We first use LexRank[1] to pre-summarize our articles. The state-of-the-art models rely on using the first two sentences of articles in order to provide the best possible summarization, here we use LexRank in order to more generalize this approach and guarantee that we are always using the most relevant sentences in our input. Next we use an additional attention model, CopyNet, in order to be able to use the source text more effectively and avoid the over usage of <UNK> and to copy salient sections of the input text directly into the summarization, such as names, dates, and rare words. With our novel ensemble of techniques, we are particularly interested in the CNN/DailyMail dataset, whose manually-written highlights we believe to be more promising than traditional headline-based summarization tasks.

1 Introduction

Text summarization is a challenge in natural language understanding that has recently evolved due to the emergence of encoder-decoder recurrent neural networks. These network models rely on sequence-to-sequence (seq2seq) models, which have been successful in areas like machine translation and speech processing. [1] At a high level, sequence-to-sequence models succeed because they take advantage of language models to predict future elements given previous elements of a sequence. The decoder uses an encoded representation of the input examples, as well as the current decoded output, to make predictions about the next outputs. These models train an encoder and decoder simultaneously.

At the moment, there exist two primary strategies for summarization: extractive and abstractive methods. Extractive summarization relies on the concatenation of extracts from some source corpus, whereas abstractive text summarization generates and paraphrases sentences that capture salient points of an article.

[1] LexRank is a non-Neural extractive summarizer based on the PageRank algorithm.

In conventional machine learnings models for summarization, researchers tend towards using articles and their headlines and input-summary pairs. However, does it really make sense to use headlines for a summarization task?

Conventionally, headlines are intended to be attention-grabbers. A successful headline encourages you to read the rest of an article. Its primary objective is not to illuminate the "big ideas" of an article.

For the machine learning task, it is generally practical to use the headline as a gold label for summarization tasks, given the large availability of datasets with news articles and their headlines. In addition, manual annotation and highlight generation is very expensive. The process requires an intentional and consistent organization by individuals with considerable expertise. As a result, many summarization/generation tasks which require large quantities of data tend towards using headlines of news articles as the label, and the article (or portions of it) as the input.

In this paper, we are primarily interested in the CNN/DailyMail dataset. This dataset is advantageous because each article comes is paired with a short set of summarized bullet points that represent meaningful "highlights" of the piece. As a result, we can be very intentional about using human-generated summaries as opposed to headlines for our summarization task.

In addition, the CNN/DailyMail corpus has been thoroughly explored in the question answering/reading comprehension task [2] to achieve state-of-the-art results in language understanding. However, this dataset has not been significantly explored for text summarization. At the moment, the few summarization papers that utilize the CNN/DailyMail dataset have achieved ROUGE-F1 scores of >35% [3]. Nallapati et al. (2016) requires an implementation with significant computational costs. Its hierarchical attention model, which achieved the highest ROUGE F1-scores, required 18 days (35 epochs) to reach convergence on a single Tesla K-40 GPU. This approach used the entire input article for training and a concatenation of all four bullet points as gold labels. Clearly, this is a significant implementation that, in many ways, reflects the upper-bound for amount of training data used.

Our goal was to explore and combine neural text summarization techniques under a more scoped version of the dataset to complete the summarization task with more reasonable computation costs. Overall, we explore various approaches to seq2seq models towards summarization that use a combination of abstractive and extractive techniques.

2 Background/Related Work

2.1 Dataset

We only use the first 2 sentence of each article in the CNN/ DailyMail dataset as training input, and the first highlight as our gold label. Intuitively, this is an effective and efficient use of the dataset, because journalists are typically trained to communicate the big ideas of an article in the first few sentences of a piece. In a foundational paper for abstractive neural summarization, the Rush et al. create input-summary pairs using the first word of each sentence and the headline of the article. [4]

Take this example:

> **First highlight:** Argentina coach Sabella believes Messi's habit of being sick during games is down to nerves.
>
> **First 2 sentences:** Argentina coach Alejandro Sabella believes Lionel Messi's habit of throwing up during games is because of nerves. The Barcelona star has vomited on the pitch during several games over the last few seasons and appeared to once again during Argentina's last warm-up match against Slovenia on Saturday.

We see that the meaning of the first highlight is largely captured in merely the first sentence – almost verbatim.

However, in some cases, the main idea is not properly or fully captured in the first two sentences:

> **First highlight:** By early Wednesday, users inside China encountered: "network timeout.
>
> **First 2 sentences:** Beijing, China has blocked the popular video-sharing web site YouTube, but did not offer a reason for the ban. YouTube was blocked in China as of Wednesday.

For a human with proper context and knowledge, it may be possible to infer "network timeout" in the gold summary from China's intentional blocking, but there is likely a need for more information.

As a result, we performed additional experiments to explore results for training our neural model on the most salient sentences in an article (as determined by using non-Neural extractive techniques) 4.2.

2.2 Abstractive text summarization

State-of-the-art models for abstractive text summarization use neural attentive encoder-decoders. Chopra et al. (2016) uses a deep convolutional neural encoder and beam-search decoder. [5] The encoder-decoder models was trained end-to-end, and used the Gigaword dataset to train on 4 million article-summary pairs. Nallapati et al. (2016) used the CNN / DailyMail dataset as a benchmark. [3]

Chopra et al. (2016) was a continuation of summarization work from Rush et al. (2015).

Nallapati et al. 2016 used a bidirectional GRU-RNN encoder, uni-directional GRU-RNN decoder with attention and the large vocabulary trick. They also ran further tests with hierarchical attention (word and sentence level attention) and pointer mechanisms.

2.3 Pointer networks

Pointer networks are a different form of attention that uses a fuzzy softmax "pointer" to take advantage of the variable-length vocabularies. The effect is that `<UNK>` tokens can be replaced by meaningful tokens from the input source text. [6]

2.4 Encoder-decoder structure

An encoder is an RNN whose purpose is to turn an input sequence into a different representation that still contains all the information about the source, but is then able to be used in different tasks. The encoder first works by taking a word and embedding it. This step allows us to capture the semantic meaning of a word and then turn it into a vector. This embedding is then input into our RNN and is turned into a hidden state. This hidden state is then used to create the next hidden state and so on until we get to the end of the source text.

An encoder has a similar structure to an encoder but its purpose is slightly different in that given a hidden state, it will try to generate a word that best corresponds to the hidden state. It can do this in multiple ways, but our model makes use of a copying attention mechanism called CopyNet that uses the source text as a way to help generate this word. The decoder also only runs until it reaches the end of our outputs, at 30 words, or if it generates an End of Sentence (`<EOS>`) token.

3 Approach

Dataset: The CNN/DailyMail dataset contains 287K documents, each with 3-4 highlights that summarize the contents of the article. We selected the first 2 sentences of each article as the input, and the first highlight as the gold label summary. The train/dev/eval split was 90/5/5.

Preprocessing: In our primary dataset, each article was processed using Penn TreeBank 3 (PTB) tokenization and lower-casing. In addition, we limited our vocabulary size to 1MM items.

In future sections (4), we will reference an entity-tagged dataset that we used for some of our experiments. This was the dataset used in the Chen et al. (2016) analysis of the reading comprehension task. All text in the entity-tagged dataset have had entity recognition and coreference resolution run.

@entity1 co-owner **@entity0** insists the club want **@entity3** to stay and sign a new contract , despite interest elsewhere .

Training: Our models were trained for >10 epochs using the following hyperparamters:

Decode: Generates output from model and corresponding label using the abstractive generation decoder or the CopyNet decoder (which allows for either copy or generation steps.

Evaluation: We run ROUGE on our generated and gold summaries for the test set. ROUGE is a common metric for automated text summarization tasks that computes a number of co-occurrence statistics between a predicted output and true output.

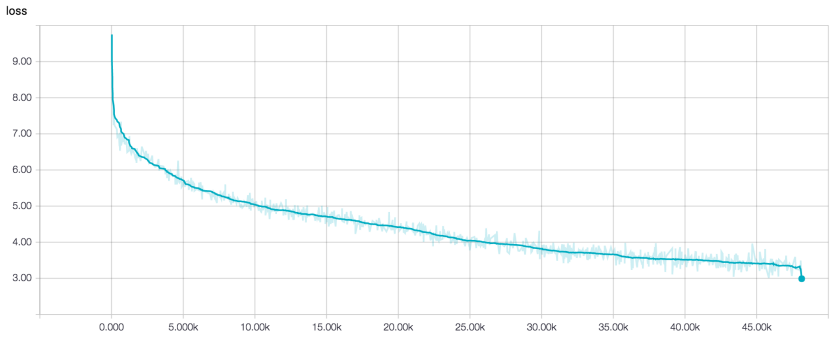

Figure 1: Loss for attentive encoder-decoder model with GloVe embeddings over 10 epochs.

4 Experiments

4.1 Baseline

For our baseline we used the open source "Textsum" Tensorflow model [2]. Textsum uses an encoder-decoder model with a bidirectional LSTM-RNN encoder and an attentive unidirectional LSTM-RNN decoder with beam search. The encoder-decoder model is trained end-to-end. For the first decode timestep, we feed in the last output of the encoder as well as the embedding for the start (<s>) token. For subsequent decode timesteps, the decoder uses the last decoder output in addition to the word embedding for the previous word to generate the next word. During the train step, the previous word is the actual previous word from the gold summary, but during the decode step, the previous word is the previously-generated word. Note that this makes the model quite sensitive to the words that the decoder has generated in the past. The decoding process will continue until the generated summary reaches the max decode length set at 30, or until it generates an EOS token.

This architecture is very similar to the attentive RNN described by Bahdanau et al., except we are using LSTM units instead of GRU units[1].

Our initial runs indicated that this model did not do well at summarization over the CNN / DailyMail dataset. After training on the CNN/ DailyMail training subset (using the first two sentences of each article as input) for 10 epochs, the decoder consistently predicted summaries with the wrong subjects. The summaries were well-formed syntactically, but they demonstrated a lack of semantic understanding of the input article. For example, it would display the word "Clinton" in the summary even though the input was clearly not about Clinton. After training on 30 epochs, we noticed a large amount of <UNK> tokens in the summaries that made them completely unreadable. Investigation on Github revealed other people who attempted to use the package ran into the same issue – it is nearly impossible to produce sensible output with Textsum unless it is trained on a much larger dataset with at least 1 million articles.

[2]Source code available here: https://github.com/tensorflow/models/tree/master/textsum

[3]Image from https://theneuralperspective.com/.

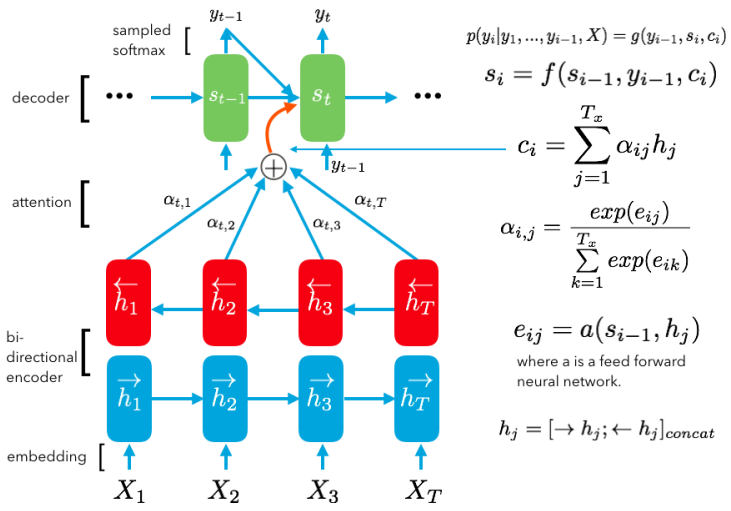

Figure 2: Attentive encoder-decoder model, where each hidden unit represents an LSTM cell.[3]

4.2 LexRank extraction

We noticed that the Textsum model only used the first two sentences of the article as input, which was standard in previous state of the art models described by Chopra et al. 2015 and Rush et al. 2016 [4] [5]. This may be intuitive since journalists are taught to summarize the idea of an article in the first two sentences. However, in many cases, we lose important information from the article when we discard the rest. To solve this problem, we first used a LexRank extraction algorithm (a non-neural extractive summarizer based on the PageRank algorithm) to find the most relevant 2 sentences to feed into our model. ROUGE-L on 2-sentence LexRank summaries was 0.247, while ROUGE-L on the first 2 sentences was 0.067, indicating that the LexRank summaries are much better initial input to feed into the model.

After running our baseline Textsum on the first two sentences of the CNN / DailyMail dataset for 10 epochs, the model consistently produced a lot of summaries for different input articles were very similar to each other. We did achieve a slight 0.001 improvement in ROUGE score, but this indicated that our language model was still dominating the output generation. As with our previous baseline, the sentences themselves made syntactic sense, but were irrelevant to the actual source input text.

Although using the two-sentence LexRank extractive summary on its own produced a higher ROUGE score than using the first two sentences of the article as the summary, feeding the LexRank output into our sequence-to-sequence models produced a much lower ROUGE score. We believe this may be due to the loss of sequential information when pulling out entire sentences from different portions of the article. Furthermore, the CNN/ DailyMail dataset seems to capture most of the early information with the first highlight (which we use as the gold summary. Later highlights typically cover information later in the article. Thus, we believe that a more complex multi-summary generation model might benefit from using LexRank to generate longer extractive summaries to effectively compress the entire article.

4.3 GloVe word vectors

We initialize our word embeddings with GloVe word vectors, which are a set of pretrained word vectors trained on the Wikipedia + Gigaword corpus of 6 billion tokens [4]Initializing our embeddings with GloVe word vectors gives our model more powerful semantic representations of the source input tokens. This initialization is especially important in our case because of the limited amount of training data in the CNN / DailyMail dataset. However, we still had unknown words at train time

[4]The pretrained GloVe word vectors that we used are available under Public Domain Dedication and License on `https://nlp.stanford.edu/projects/glove/`.

that are not properly embedded because news articles tend to describe very specific events and entity names that didn't appear in the GloVe vocabulary. Take the example:

> **First 2 sentences:** A 61-year-old soldier says he has been kicked out of the army because he is 'too old' - even though he can still run a mile-and-a-half in just ten minutes. Fitness fanatic Kevin Fulthorpe spent 25 years in the army reserve, where he gave fitness training to his younger comrades.
> **First highlight:** kevin fulthorpe , from cardiff , was part of the army reserve for 25 years
> **Summary output:** kevin <UNK> has been kicked out of army

Analyzing our vocabulary revealed that 70% of our training vocabulary exists in the GloVe vocabulary, which indicates decent vocabulary coverage, but also highlights a possible area for improvement.

4.4 Copying attention mechanism

The copying attention mechanism is similar to the original sequence to sequence attentive mechanism described by Bahdanau et al. However, instead of simply using the encoder-generated hidden states as an additional input to the decoder, we create a probabilistic model to predict whether to use the hidden state to generate a word from the training vocabulary or to use the hidden state as positional information to copy a word from the article instance-specific source vocabulary. In the generate mode, the model uses each hidden state as in the original sequence to sequence attentive mechanism. In the copy mode, the model uses each hidden state as information about the word at the corresponding positional index (so hidden state 1 is used as a representation for input word at position).

When determining the mode and the next word in the summary, we want to consider the probabilities of all possible words that appear in the union of the training vocabulary set and the source vocabulary set. The probabilities for each (word, mode) pair are calculated then combined to get the probability of each word.

The probabiliites generated by P(gl) is given by equations

$$\psi_g = v_c^T W_o s_t$$

Where v_c and W_c are parameters that are being learned similar to word embeddings. The scoring function for the copy probabilities is given by

$$\psi_c = \sigma(W_c h_j) s_t$$

Where h_j is the hidden state corresponding to the jth word, s_t, is the hidden decode state at time t, and w_c is a parameter.

The idea of CopyNet is similar to Pointer Networks in that both architectures allow the models to take advantage of the instance-specific source text (described in Gulcehre et al. and used in Nallapati et al.). However, Pointer Networks don't combine the probabilities of words across the the source and generating a word form the vocabulary, a target word is chosen from the entire probability distribution over both generate and copy modes as follows.

$$p(y_t|s_t, c_t, M, y_{t-1}) = p(y_t, g|s_t, c_t, M, y_{t-1}) + p(y_t, c|s_t, c_t, M, y_{t-1}$$

Because we now have a chance to copy a word from the source, we need to embed that word, so we represent the previous word with a concatenation of the word embeddings and a weighted average of the hidden states from the source text, $y_{t-1} = [e(y_{t-1}); \zeta(y_{t-1})]$. Here $e(y_{t-1})$ is given by the word embedding for word y_{t-1}, which if the previous word was not in the vocab, is just 0s. Similar to the context vector,

$$\zeta = \sum_j^J \beta_{tj} h_j$$

[5]Image from [7].

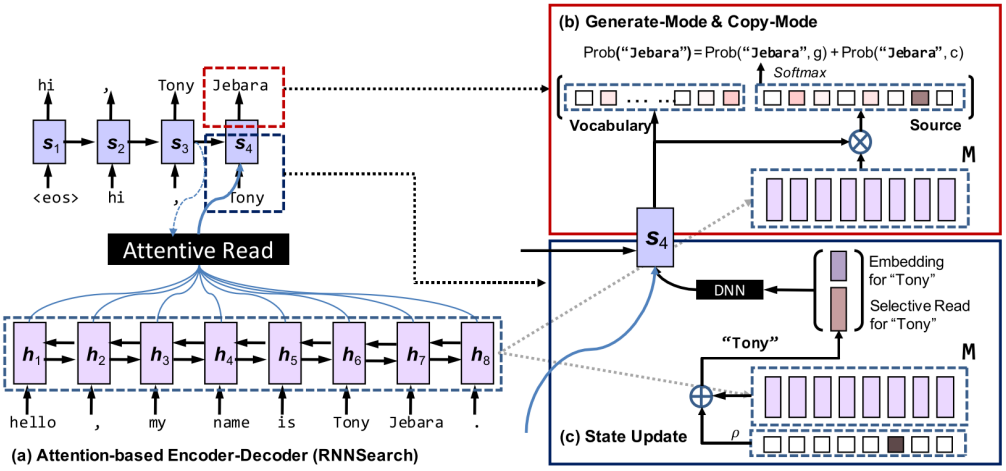

Figure 3: Copynet: architecture for copy and generation modes. [5]

$$\beta_t j = \frac{e^{\psi_c(h_j, s_{t-1})}}{\sum_k \psi_c(h_k, s_{t-1})}$$

Where h_j is the jth hidden state in the source text, and $p(c|)_j$ is the probability for producing This allows us to also update the next hidden state with this new "embedding" and allows for the decode step to learn from the source text itself as well as use semantic meaning not only from our word embeddings, but directly from the source which makes it more powerful as we have the meaning of the word in the context in which it is defined. This is the main difference We generate a new state by using the LSTM by feeding it in the $y_t - 1$, $h_t - 1$, and c_t.

By training our new model end to end with this new copying attention mechanism, the network learned the appropriate parameters in order to use the encoder hidden state and chose whether to generate or copy the next word, and then chose the right word out of the source vocab or the training vocab.

4.5 Results

	Preprocessing	Learning model	ROUGE-1	ROUGE-2	ROUGE-L
1	None	Textsum	0.08600	0.00771	0.07875
2	Tokenized	Textsum	0.08154	0.00941	0.07623
3	Tokenized	Textsum w/ Copy	0.04869	0.00465	0.04494
4	Entity-tagged	Texsum	0.06434	0.00380	0.05730
5	Entity-tagged	Textsum + GloVE	0.04035	0.01290	0.03820
6	**Tokenized**	**Texsum + GloVe**	**0.12123**	0.03604	**0.11342**

Generally, tokenization did not produce noticeable gains, potentially because it increased the number of tokens in our source and summary texts, while we kept the number of encoding timesteps (i.e. input length) at 120 tokens. We truncated any articles and gold summaries that exceeded this length, which may have caused us to lose some information in certain examples.

The model with the copying mechanism (experiment 3 in the table) was trained on a very small dataset of about 10,000 articles until the loss was below 0.01 as an initial experiment. Unfortunately we were not able to train a full 10 epochs in time to produce meaningful decoding results. The training time is much greater because of the increased number of parameters we needed to train. However, on our small experimental subset of the dataset, the copying mechanism did successfully copy proper nouns and other tokens from the source texts. This gave the summaries an appropriate

subject (whereas our baseline attentive sequence to sequence summarizer often missed the correct main subject in the summary). For example:

> **First 2 sentences:** Police have uncovered a £1million 'Downton Abbey' mansion which had been transformed into a cannabis factory containing more than 1,000 plants. Wendreda House in March, Cambridgeshire, was raided by officers yesterday after an anonymous tip-off about a pungent smell coming from the three-storey property.
>
> **First highlight:** wendreda house was raided by officers after a pungent smell was reported
>
> **Copying mechanism summary:** 1,000 1,000 plants wendreda house march march was by raided raided was , , march in wendreda wendreda plants plants plants plants plants plants plants plants plants plants plants
>
> **Textsum summary** manor ' mansion was raided by officers

The copying mechanism summary shows many repeated words in sentences, something not seen in other models. We believe this to be in large part due to a lack of training time. After the few epochs that we were able to train on both the small experimental training set and the full training set, the language model was not fully trained, and the copy mechanism seemed to be dominating the output generation.

Our best results came from our 6th experiment, as we combined a tokenized dataset and the pre-trained embedding generated from GloVe. We think this model did the best because it solved the problem of using such a small training set. Our model greatly benefitted from word embeddings that had been trained on a much larger corpus. This can easily be seem from our examples with this model as the words chosen are pertinent and salient to the article. However, we saw an increased amount of <UNK> tokens in this model.

5 Conclusion

We introduced a unique combination of copying attention and generation to an encoder-decoder model with a bidirectional LSTM-RNN encoder and RNN decoder. We completely integrated this new attention mechanism into the existing complex Textsum architecture. Adding GloVe embeddings helped give semantic meaning to the input words, which helped improve the relevance of the output summaries to the corresponding input articles.

Adding a copying attention mechanism to augment our attentive generation mechanism greatly reduced the number o2f <UNK> tokens that were generated by our model.

Finally we realize that the quality of our summaries often did not correlate strongly with their assigned Rouge scores. As Rouge uses unigram and bigram counts to evaluate the summary, this makes it hard to mimic any human summary in news articles because what is relevant can differ greatly for people, especially in longer and longer articles such as the ones found in the CNN/ Dailymail dataset. We could always see a significant difference between our unigram and bigram scores as its easier to pick up on the important words in the document, but because of the nature of abstractive summaries, what goes in between can be anything that makes sense for that language, ending with very low Rouge-2 scores even when the summary seems perfectly acceptable.

Acknowledgments

We'd like to extend a special thank you to our project advisor, Danqi Chen, who provided wonderful guidance and patience throughout the process.

In addition, thank you to Richard Socher, Chris Manning, and the CS224N course TAs for their office hours and dedication to creating a rewarding learning experience.

References

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation By Jointly Learning To Align and Translate. *Iclr* 2015, 1–15.

[2] Chen, D., Bolton, J., & Manning, C. D. (2016). A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. *Acl* 2016, 2358–2367.

[3] Nallapati, R., Zhou, B., Santos, C. N. dos, Gulcehre, C., & Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Proceedings of CoNLL*, 280–290.

[4] Rush, A. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (September), 379–389.

[5] Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. *Naacl-2016*, 93–98.

[6] Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., & Bengio, Y. (2016). Pointing the Unknown Words. *Acl 2016*, 140–149.

[7] Gu, J., Lu, Z., Li, H., & Li, V. O. K. (2016). Incorporating Copying Mechanism in Sequence-to-Sequence Learning. *Acl, 11*.