Sentence-Level Extractive Text Summarization

Nathan Zhao, Yijun Jiang, Yi Liu
{nzz2102, yijunj, yiliu021}@stanford.edu

Abstract

In this project, we aim to reproduce the NeuSum network, which extracts a fixed number of sentences from documents, and train it on a different dataset from the original. We implemented our own NeuSum model from scratch, then trained and tested it on a sentence-level extractive dataset which we preprocessed from the Newsroom dataset. We also discuss an upgraded version of NeuSum ("adaptive NeuSum") that generates extractive summaries with adaptive length.

1 Introduction

Text summarization is an important natural language processing task which compresses the information of a potentially long document into a compact, fluent form. Research has been conducted in two types of text summarization: extractive and abstractive. Extractive summarization seeks to select a subset of the words or sentences in the existing document which best represents a summary of the document. Abstractive techniques attempt to generate an entirely novel reconstruction of a summary. Historically, both tasks have been relatively difficult, even for neural approaches.

Traditional approaches to text summarization focuses on extractive techniques at sentence level. With neural network, the summarization process is usually divided into two parts: sentence scoring and sentence selection. Sentence scoring aims to score each sentence, and sentence selection chooses sentences with strategies according the sentence scores. While most work separately deals with the two parts, we noticed a model called NeuSum (**Neu**ral extractive document **Sum**marization) [1] that integrates sentence scoring and selection and jointly trains the two neural networks. It even achieved higher Rouge scores compared with the simple yet powerful LEAD-3 models and many other models. Therefore, we chose this work as our main reference and re-implemented the whole model by ourselves. We trained and evaluated this model on a different dataset (Newsroom [2]) but found that it does not outperform LEAD-3. In addition, considering that NeuSum is only capable of generating summarizations with fixed number of sentences (3 sentences in the publication), we introduce a modification to the model so that it can adaptively output summaries with different lengths.

2 Related Work

NeuSum is the first end-to-end neural architecture of its kind. Nevertheless, there have been several algorithmic approaches to extractive summarization. The simplest is the LEAD-3 benchmark, which proposes the first three sentences of the document as the summary. It is the best extractive baselines (as of Jan. 2018) for the Newsroom dataset with an average Rouge-1 F1 score of 30% [3]. TextRank [4] is an unsupervised learning algorithm which extracts keyphrases. Its benchmark score on Newsroom is 20% and represents the best non-brute force or LEAD-3 type benchmark on the Newsroom dataset. Other types of extractive summarization include graph based approaches [5], bayesian topic models [6], etc.

The NeuSum architecture has not been benchmarked on this dataset but has consistently achieved much higher Rouge scores when applied to the CNN/Daily mail dataset against other state of the

Final report of Stanford CS224N (Natural Language Processing) final project in Winter 2019.

art methods [1]. Considering the fact that the CNN/Daily mail dataset and the Newsroom dataset have different distributions, we think it worthwhile to evaluate the performance of NeuSum on the Newsroom dataset.

3 Approach

Although we are aware that the original paper has a PyTorch implementation, **we did not use it but implemented the architecture from scratch** so that the structure is clearer and the codes are more understandable. Our code can be found on Github: `https://github.com/yijunj/CS224N_Summarization`. We did refer to Assignment 4 for the overall structure of an NLP model. Our model is described in detail as the following.

3.1 NeuSum Architecture

As is shown in Fig. 1, the model is composed of a sentence-level encoder, a document-level encoder, and a joint sentence scoring and selection part. The sentence-level encoder reads in each sentence word-by word with a bidirectional GRU and concatenates the last forward and backward GRU hidden vectors as the sentence-level representation of the sentence. These representations of sentences in a document are then sent into the document-level encoder (also a bidirectional GRU) to generate the document-level representations of the sentences. Next they are used as inputs to the joint sentence scorer and selector. At each time step t, the joint sentence scorer and selector calculates the scores $\delta_t(S_i)$ of all candidate sentences S_i with a GRU cell and a multi-layer perceptron, and selects the sentence with the highest score as the next sentence in the output summary. The representation of the selected sentence then becomes the input to the GRU cell to produce the updated hidden state and therefore the updates sentence score for the selection of the next sentence. This process terminates when the desired number of sentences has been selected.

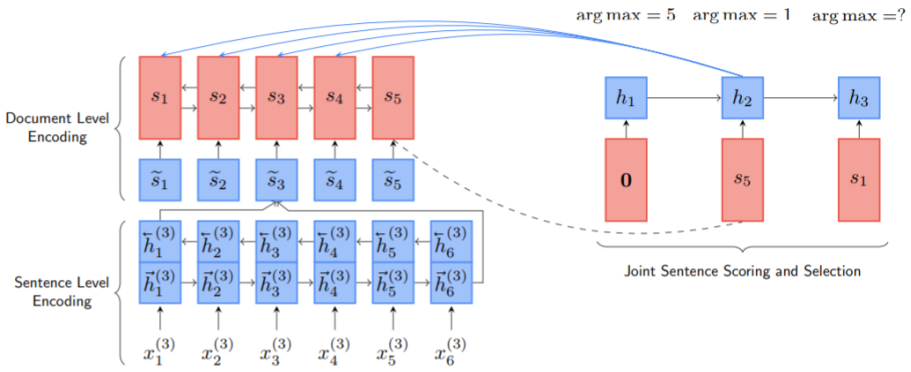

Figure 1: NeuSum architecture schematics. Figure comes from the original paper [1].

3.2 Evaluation Method

The loss function used in the NeuSum model is based on the Kullback-Leiber (KL) divergence of the model's predicted distribution of the sentence scores and the labeled training data distribution. This loss is required for every time step t of sentence selection (in practice it is computed with a sequence of all-time-step outputs, but only after the very last time step is made). The sentence scores $\delta_t(S_i)$ at time step t are normalized via a softmax such that

$$P_t(S_i) = \frac{\exp(\delta_t(S_i))}{\sum_{k=1}^{L} \exp(\delta_t(S_k))} \tag{1}$$

To calculate the labeled training data distribution, we use a softmax operation with a temperature parameter τ set to 20

$$Q_t(S_i) = \frac{\exp(\tau g_t(S_i))}{\sum_{k=1}^{L} \exp(\tau g_t(S_k))} \tag{2}$$

where $g_t(S_i)$ is the normalized incremental Rouge-1 F1 score of adding sentence S_i to the partially extracted summary at time t. Finally we calculate the KL-divergence loss function

$$D_{KL}(P_t||Q_t) = -\sum_S P_t(S) \log\left(\frac{Q_t(S)}{P_t(S)}\right) \tag{3}$$

and sum over all time steps to get the final loss. We implemented this objective function by ourselves, with the help of `torch.nn.KLDivLoss`. Unfortunately, this loss involves evaluating the Rouge score of **all possible extractions given the current summary**, and **over all time steps**. It is in practice extremely challenging to make this loss computation fully vectorized and batch-compatible, and `for` loops are inevitable in the actual code. This greatly limited the speed of our training. Consequently, the largest dataset we trained on contained only 10,000 samples. A training of 200 epochs could take up to 11 hours on an Azure NV6 VM and we were surprised to find that Azure NV12 VM didn't make the training process any faster.

To evaluate the trained model on the test dataset, we calculated the Rouge-1 F1 score against an artificially extracted reference summary, which is described in the next section. We then compared the Rouge scores with the LEAD-3 baseline.

4 Experiments

4.1 Data

Our dataset is the Newsroom dataset, a collection of 1.3 million documents and concurrent summaries written by authors and editors from 38 major news publications. Balancing between the quality of model and the training time, we subsampled approximately 10,000 of these documents, and preprocessed them as is discussed below. We also created a toy dataset made up of two documents using a tiny vocabulary of 12 words. This allowed us to quickly build up and test various submodels in the NeuSum architecture.

4.1.1 Data Preprocessing

We first broke each document into sentences. This requires more work than naively splitting strings at full stops (and question/exclamation marks), as many abbreviations contains periods that do not signify the end of a sentence. We had to hand-craft some regex patterns to filter out true sentence endings. We then split each sentence into words and built a vocabulary using the entire corpus. With this vocabulary we tokenize all documents and summaries and padded each batch to the same length (both document-wise and sentence-wise). The average document length is 50 sentences long and the average sentence is 19 words long.

4.1.2 Generation of Sentence-Level Extractive Reference Summaries

Since the reference summaries in the Newsroom dataset are not intrinsically sentence-level extractive, we followed the same approach as the NeuSum paper and artificially extracted sentences as summaries for each document. One of the first challenges, as can be seen in Fig. 2c, is that even if we analyze all the documents word for word, only around $40 - 50\%$ have a reasonable amount of word overlap with their gold summary, which partially motivates why extractively summarizing may be more appropriate.

The original NeuSum paper followed a greedy approach, probing different combinations of sentences to maximize the Rouge-1 F1 score with respect to the original reference summaries. However, given that the average document length in our corpus is over 50 sentences, probing all combinations of 3 sentences would mean 19,600 combinations. In order to speed up their approach, we modified it to cap the number of sentences it could search to the first N where we select $N < 50$. Even with conservative figures of $N = 10$ and searching combinations of 2-4 sentences in these first 10, we could achieve optimized Rouge scores of over 0.45.

In order to avoid the huge computational load of the greedy, semi-exhaustive search method mentioned above, we came up with a significantly faster method. Instead of a brute force probe of combinations, we evaluated the Rouge-1 metric for each sentence of the document with respect to the original reference summary. From there, we took the k-highest scoring sentences where k was chosen to be

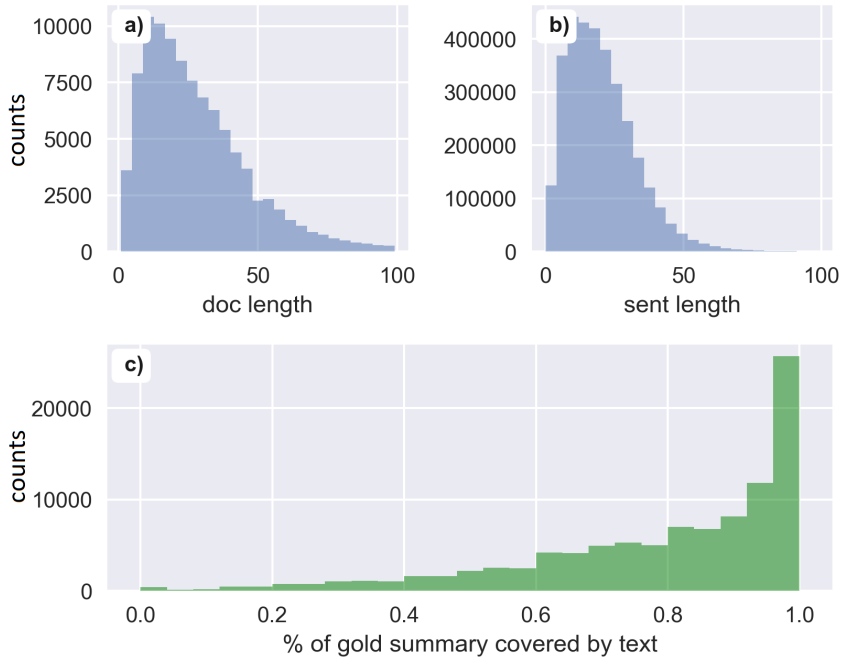

Figure 2: a) distribution of document length (number of sentences per document in the newsroom dataset). b) distribution of sentence length (number of words per sentence) over the corpus. c) information content overlap of the abstractive gold summaries versus the text measured as the percentage of the summary that can be constructed by words found in the document.

in proportion to the length of the text. In the end, our optimized k-best sentence selector produced a Rouge-1 F1 score of 0.397, which is an improvement of over 0.1 from the LEAD-3 baseline. However, based on the distributions shown in Fig. 3 d) and e) it does appear to overselect the first two sentences relative to an exhaustive search, which is non-ideal as this pattern can be easily learned by extractive learning algorithms.

To combine the best of both worlds, the final algorithm we propose is effectively a forward-selection algorithm. Essentially, we started out with an empty summary and found the first sentence which maximizes the Rouge score. After that, we can then add in the next sentence which best maximizes the Rouge score. Of course, adding in an additional sentence may not increase the previous best Rouge score, so we can either truncate the summary there or add the sentence which decreases the score the least. In the latter case, we continue this process up to some maximum summary length in order to ensure that the extractions all have the same length. With this method we could achieve a maximum Rouge score with the reference of 0.50 (higher than both the modified exhaustive and the k-best methods) as well as achieving a much more balanced distribution of sentences selected for the final summary. In practice, we chose $k = 20$ to balance between a wide enough sentence index distribution and an acceptable runtime.

During reference summary extraction, one of the interesting things we found is that the forward selection algorithm typically yields superior Rouge scores if we allow variable length summaries rather than forcing the summaries to all be of a constant length. The forward selection algorithm can be designed so that if adding the i-th sentence to the current summary cannot increase the Rouge score for any choice of the i-th sentence, then we do not attempt to further build the summary. Interestingly enough, we note that this multi-length rejective forward selection method has a distinct advantage over a fixed-length method. From Fig. 4, the multi-length selection achieves a Rouge-1 F1 score of 0.56 while the 3-sentence forward selection achieves a Rouge-1 F1 score of just 0.5. This inspires us to design an adaptive NeuSum model that learns the optimal length of the generated summary. We will discuss adaptive NeuSum towards the end of this report.

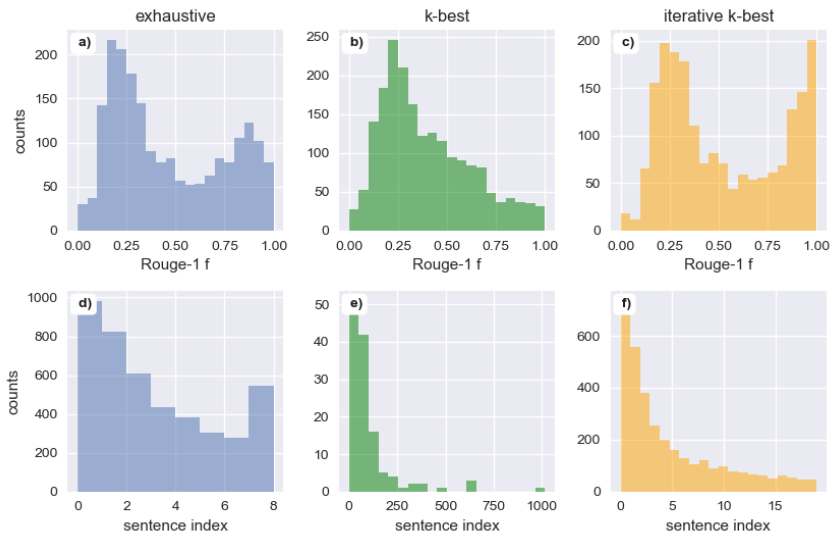

Figure 3: Statistics based on sentence-level extractive reference summary generation over 2,000 documents. Top row shows the distribution of Rouge-1 F1 scores, and bottom row shows the index distribution of sentences that are selected as summary. Note that in k-best all sentences in a document are explored, whose indices occasionally go up to above 1,000, while in the other two algorithms we have only looked at sentence indices no more than 10 (or 20), for they run slower than k-best. The last column titled "iterative k-best" refers to the forward selection algorithm with multi-length output.

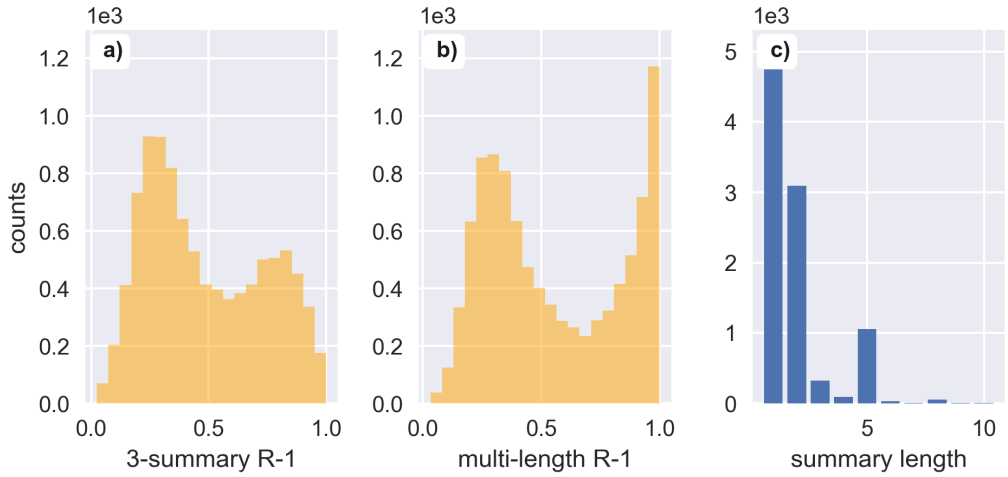

Figure 4: a) distribution of extracted summary Rouge-1 F1 scores of forced 3-sentence extracted summaries with respect to the abstractive summaries in Newsroom. b) distribution of extracted Rouge-1 F1 scores allowing for variable length summaries. c) Distribution of summary lengths in b). Note that the first bar is for summaries of length 1.

4.2 Training Details

Several tricks have been done to make full use of batch processing to improve efficiency. For instance, the input documents has shape `(batch_size, document_length, sentence_length)`, which needs to be reshaped into shape `(batch_size*document_length, sentence_length)` before sending into the sentence-level encoder. The output of the sentence-level encoder is reshaped once

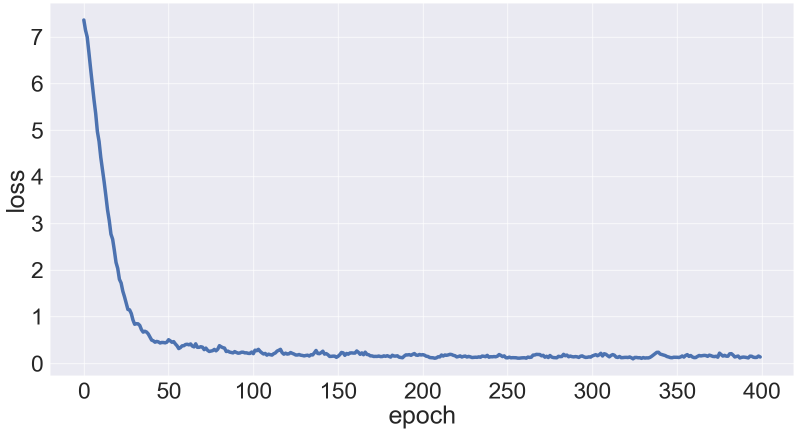

Figure 5: (Per-document) loss curve when trained with 1,000 documents and 3-sentence extractive reference summaries.

more to recover the original dimensions before feeding into the document-level encoder. In the joint sentence scorer and selector, similar reshape operations are used to facilitate batch processing.

Following the original NeuSum paper, we chose the size of word embedding, sentence level encoder GRU, document level encoder GRU to be 50, 256, and 256, respectively. We also set the sentence extractor GRU hidden size to 256.

During the training process, we chose a batch size of 8 documents in order to fit into the GPU memory. Document length was capped at 20, and each document had been padded to the longest document in the same batch. All sentences in a batch were also padded to the same length. We used the Adam optimizer with $\alpha = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The dropout probability was 0.3 after the sentence-level encoder and 0.2 after the document-level encoder.

As a starting point, we trained it on only 10 documents. We could easily overfit the dataset with the network, which proved the correctness of our model implementation. Then we trained on actual datasets of 1,000 to 10,000 documents. Each document has an original reference summary, as well as a generated extractive reference summary, which will be used as the ground truth. We prepared both datasets with 2-sentence and 3-sentence extractive reference summaries.

4.3 Results

Here we refer to NeuSum trained with n-sentence reference summaries as the n-summary model, where $n = 2$ or 3 in our implementation. Their behaviors during training are similar, though a 3-summary model is 50% more time-expensive. Fig. 5 shows the convergence of loss for a 3-summary model.

The Rouge scores of the n-summary models as well as the LEAD-n baselines are summarized in Table 1. On the Newsroom dataset, the NeuSum model achieves Rouge-1 F1 scores around 40% for 2-sentence and 3-sentence extraction tasks, which is about 10% lower than the performance of LEAD-2 or LEAD-3. Notice that these Rouge scores are considerably higher than the published benchmark scores on the Newsroom dataset website (30% for LEAD-3). This is because our Rouge scores are calculated against the artificially extracted summaries but not the original abstractive summaries.

Table 1: Rouge-1 F1 scores (%) of LEAD-n baselines and n-summary models ($n = 2$ or 3)

Test Dataset	LEAD-2	LEAD-3	2-summary Model	3-summary Model
2-Sent Extraction	49.7	–	38.7	39.4
3-Sent Extraction	–	53.7	40.6	41.7

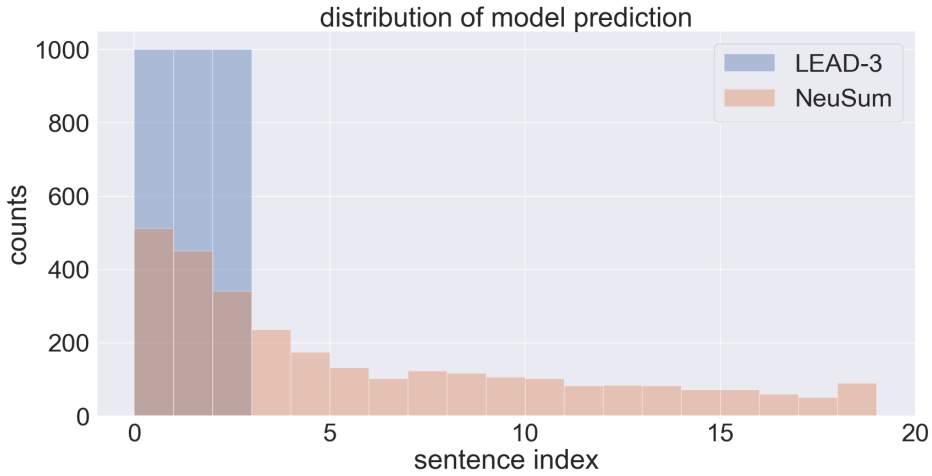

Figure 6: Distribution of sentence indices predicted by the trained NeuSum network as well as the LEAD-3 baseline. In contrast to LEAD-3, the trained model shows a long tail into large-index sentences, which is in agreement with the extractive reference generator distribution in Fig. 3f.

5 Analysis

5.1 Comparison between our NeuSum model and the original NeuSum model

As shown in Table. 1, we concluded that LEAD-3 and LEAD-2 outperform the NeuSum model on the Newsroom dataset, which is different from the conclusion in the original NeuSum paper that used the CNN/Daily Mail dataset. This is probably because of the different distributions of Newsroom and CNN/Daily Mail dataset. In addition, while we only trained our model on 10,000 documents, the original NeuSum maodel was trained on 287,227 documents. Furthermore, we updated our word embeddings during training while the original paper used the pretrained GloVe word embeddings. These factors could result in the difference in model performances.

5.2 Rouge-*n* Scores and Sentence Ordering

As mentioned above, we ran an overfitting test with a small dataset to validate that we are correctly implementing the model at early stages. During this test, we observed something interesting: as the loss decreases during the training process, the model learned to pick the right sentences, but it did not learn the order of selected sentence in the output summaries. This can be attributed to the usage of the Rouge-1 F1 score for loss calculation - it only considers one gram. However, when we used Rouge-4 F1 score (four grams) in the loss calculation, the model correctly overfitted the order of sentences as well, which met our expectation. Considering that the NeuSum paper also used Rouge-1 F1 scores, we propose that using Rouge-n F1 scores ($n \geq 2$) will help the model learn the sentence ordering information as well, though we lacked the time to try it out on large dataset.

5.3 Distribution of Selected Sentences

As shown in Fig. 6, we calculated the distribution of sentence indices selected in summaries by the NeuSum model. Compared with the distribution of sentence indices for LEAD-3, the distribution for NeuSum features a long tail and is more scattered over sentence indices. Similarly, the tail feature also exists in the reference summaries (Fig. 3f). This similarity can be attributed to the model learning the distribution of the reference summaries in the dataset. Although LEAD-3 earns a higher Rough-1 F1 score than the NeuSum model, the latter is able to catch the main feature of the distribution of dataset. This feature-capturing property was also validated in the original NeuSum paper.

5.4 Adaptive Summary Length with NeuSum

To design an adaptive-length NeuSum model, we notice that in practice padding sentences do not count into the Rouge scores. We also assume that when the summary reaches optimal length, appending to it any other non-padding sentence will lower the Rouge, leading to a higher loss. Thus, simply by allowing the model to sample padding sentences as potential candidates, we give the model freedom not to choose any actual sentences once the generated summary has no room for improvement. The model still performs a fixed number of sentence selection steps, yet by removing padding sentences from the final output, the length of the generated summary becomes adaptive.

To confirm this, we train the NeuSum model on a 2-sentence summary dataset, but set the number of selection steps to be 3. It can be seen in Fig. 7 that indeed as the model is trained, there is increasing chance that it outputs a padding sentence, i.e. it recognizes that the ground truth is in fact of length 2.

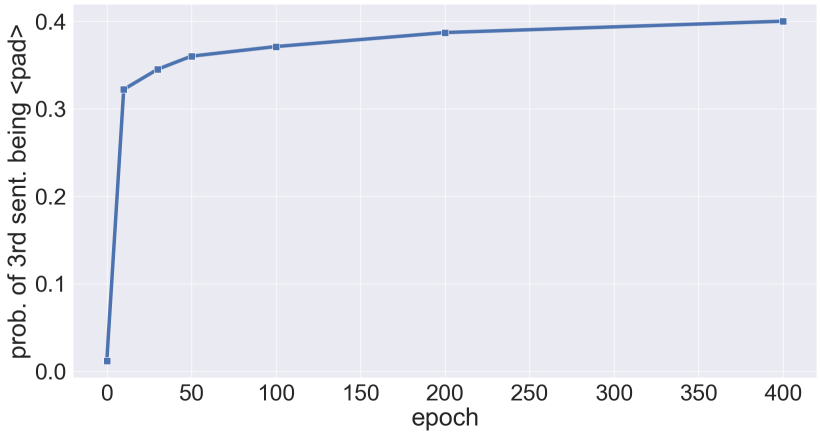

Figure 7: Probability of the model selecting a padding sentence as the 3rd in the output summary, when trained on a dataset of 2-sentence reference summaries. Clearly the model is learning the correct summary length over time.

6 Conclusions and Future Work

In this project, we implemented our own NeuSum model from scratch and trained and evaluated it on a sentence-level extractive dataset after preprocessing the Newsroom dataset. Unlike the original NeuSum paper, we found that our NeuSum model was outperformed by LEAD-2 and LEAD-3 due to several possible reasons, but it managed to learn the distribution of the dataset. Furthermore, we discussed the adaptive NeuSum and found that our NeuSum model is potentially capable of deciding the length of the summaries by itself, which is an upgrade compared with the orignal NeuSum model that selects a fixed number of 3 sentences as summaries.

We have envisioned some possible future work. First, it is worthwhile to train the NeuSum model on a larger Newsroom dataset for a even longer time to better evaluate its performance on this dataset. Secondly, it would be interesting to implement a real adaptive NeuSum model. For example, if we would like to generate summaries with lengths $\leq n$, we would 1) pad the longest document in the training dataset with $n-1$ more <end> token sentences and also pad other documents to the same length with <end> sentences and 2) set the model to perform n sentence selection steps while allowing the selection of <end> sentences. As discussed before, a well-trained model should be able to generate summaries with optimal lengths $\leq n$.

7 Acknowledgement

We would like to thank the entire CS224n teaching team, especially the project mentor Anand Dhoot for valuable feedback during our work on this final project.

References

[1] Q. Zhou *et al.*, *Neural Document Summarization by Jointly Learning to Scoreand Select Sentences*, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pp. 654–663 (2018)

[2] M. Grusky, M. Naaman, and Y. Artzi, *Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies*, arXiv:1804.11283 (2018)

[3] Newsroom dataset website: `https://summari.es/` (accessed 03/2019)

[4] R. Mihalcea and P. Tarau, *Textrank: Bringing order into texts*, Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–411 (2004)

[5] Thakkar *et al. Graph-Based Algorithms for Text Summarization*,2010 3rd International Conference on Emerging Trends in Engineering and Technology, pp. 516-519 (2010)

[6] Allahyari*et al. Text Summarization Techniques: A Brief Survey*, arxiv (2017)