Neural Text Summarization

Urvashi Khandelwal*
Department of Computer Science
Stanford University
urvashik@stanford.edu

Abstract

Generation based text summarization is a hard task and recent deep learning attempts show that sequence to sequence models hold promise. In this paper, we approach this problem with a similar class of models, in a relatively smaller data setting and attempt to alleviate the challenges faced with imitation learning.

1 Introduction

Modern deep learning architectures have attained competitive performance on many tasks relating to natural language. In particular, sequence to sequence models have proven useful for machine translation, speech recognition, image captioning and many others. In this paper, we adapt these models to tackle text summarization which involves generating a short summary for a longer piece of text. More specifically, we attempt to generate paper titles for scientific publications given their abstracts.

Traditional text summarization techniques have often relied on extracting features from the input text and using these as inputs to naive-bayes classifiers (Kupiec et. al., 1995; Aone et. al., 1999) or log-linear models (Osborne, 2002), making per sentence decisions on whether the sentence is a candidate for the summary. These are predominantly extractive summarization techniques. Abstractive summarization, on the other hand, selects a sentence generated using the target vocabulary. Traditional summarization techniques have relied on manually designed features such as $tf - idf$ scores, or positional information, which is often an argument in favor of deep learning techniques since they learn which features are important automatically.

Recently, two groups have worked on abstractive text summarization using deep neural networks (Rush et. al., 2015, Nallapati et. al., 2016). Both of them used the encoder-decoder paradigms to summarize news articles. In this paper, we design similar models to summarize NLP publication abstracts, using their titles as ground truth. We use imitation learning techniques to push the training regime to match the evaluation setting, in the hopes of addressing the question of whether this helps the model learn better.

The rest of the paper is organized as follows: in Section 2 we discuss some background for the task, in Section 3 we dive into the related work. Then we detail our model in Section 4, describe and analyze data and experiments in Section 5 and conclude with a discussion on future work in Section 6.

2 Background

We first define the text summarization task. Given an input sequnce $x = \{x_1, x_2, \ldots, x_n\}$, we would like to generate an output sequence $y = \{y_1, y_2, \ldots, y_m\}$ which summarizes the input. We work with abstractive summarization and treat the input vocabulary, $\mathcal{V}_\mathcal{X}$, as separate from the output

*Joint work with Peng Qi and Dan Jurafsky

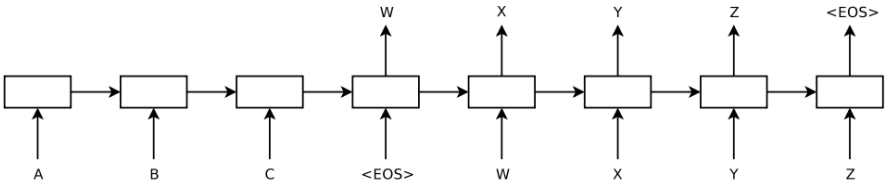


Figure 1: Sequence to Sequence model (image from Sutskever et. al (2014)). $x = ABC$ and $y = WXYZ$ and the decoder is initialized with h_x, the final hidden representation of the encoder.

vocabulary, $\mathcal{V}_\mathcal{Y}$. In abstractive summarization, summaries are generated using the target vocabulary $\mathcal{V}_\mathcal{Y}$, i.e. given the set of summaries of maximum possible length m, we select the best one under a scoring function s:

$$\arg\max_{\hat{y}} s(x, \hat{y}) \tag{1}$$

This is in contrast to extractive summarization where the output is generated using sentences from the input. In this paper, the scoring function is synonymous to the negative of the model's loss function, as described in Section 4.

3 Related Work

For the purposes of this report, we focus only on the neural summarization models by Rush et. al. (2015) and Nallapati et. al. (2016). They both operate on the Gigaword Corpus containing news articles. After preprocessing they use 4 million articles, which is 2 orders of magnitude larger than the corpus used in this study.

The model in Rush et. al. is an encoder-decoder model where the encoder is a convolutional network and the decoder is a feedforward neural network language model. They integrate attention into the convolutional encoder and end up using the trained neural network as a feature to a log-linear model, similar to what Cho et. al. (2014) did for their machine translation encoder-decoder model. Using a convolutional encoder means that they used a bag of n-grams model, ignoring the overall sequence order while generating a hidden representation. They only used the first sentence of each news article to generate its title. From an analysis of our dataset, we know that this will not be enough since unigram overlap between the title and abstract is higher with more sentences.

Nallapati et. al. build an encoder-decoder model using LSTMs and augmented their model with hierarchical encoders and hierarchical attention which learns word and sentence level attention. They operated on the same dataset as Rush et. al. but used the first two to five sentences, reporting improved performance with two sentences. Both these models capture interesting pieces of summarization as a generative task. However, both follow different training and evaluation decoding methods. They use gold tokens at training time for each timestep and greedy/beam decoding at evalutaion. Since they have a large amount of data, this disparity might not be amplified, but with a smaller dataset, as in our case, we can demonstrate the problems with this.

4 Model

In this section, we define the model used to address the text summarization task. We detail the setup of the RNN based encoder and decoder. We also provide some motivation and detail for the use of imitation learning.

4.1 Encoder-Decoder models

In a vanilla sequence to sequence model, as described by Sutskever et. al. (2014) and illustrated in Figure 1, we have two components: (1) an encoder which reads in the input sequence $x \in \mathbb{R}^n$ and computes a hidden state representation, h_x for it, (2) a decoder which uses h_x to generate the target

sequence $y \in \mathbb{R}^m$. The goal of this model is to compute the conditional probability distribution:

$$p(y|x) = \prod_{i=1}^{m} p(y_i|y_{<i}, h_x) \tag{2}$$

where each term in the product is the output of a softmax over the target vocabulary. In our model, both the encoder and the decoder are Long Short Term Memory (LSTM) networks. These were primarily chosen to capture the long term dependencies between the input and output as well as to alleviate the vanishing gradient problem. For text summarization, the input sequence is often longer than the output sequence. We leave the reader to consult the details for LSTM models as described by Graves (2013).

We optimize the model using simple SGD, minimizing the cross entropy loss:

$$CE(y, \hat{y}) = -\sum_{i=1}^{|V|} y_i \log \hat{y}_i \tag{3}$$

where $\hat{y}$ is the output at the decoder softmaxes and y is the target sequence. Errors backpropagate from the decoder into the encoder through h_x.

4.2 Imitation Learning

The simple sequence to sequence model suffers from divergence in the train and test data distributions. This happens because at training, the model is provided with the previous gold token at each timestep, while at test time, since the gold token is not available, the previous timestep's prediction is used as input. This problem is especially amplified when the training data is small, as in our case. In Section 5, we provide some analysis to this effect. Hence, we utilize an imitation learning technique - Data-as-Demonstrator (DaD) - as described by Venkatraman et. al. (2015), to resolve the issue.

In this technique, we modify the training regime to also use the previous timestep's output $\hat{y}_{t-1}$, just as in the testing regime. This is done by flipping a coin at each timestep to decide between y_{t-1} and $\hat{y}_{t-1}$. The loss is computed in the same way as before. This allows the model to learn by using the data as a corrector with the perfect policy, as opposed to being entirely supervised. The input at the first timestep of the decoder is always the special <EOS> token.

5 Experiments

Now we describe the dataset used, provide some implementation specific details such as hyperparameters, report experimental results and offer an analysis of these results.

5.1 Dataset

The dataset used in this project was the ACL Anthology Reference Corpus (Bird et. al., 2008). It is a collection of all publications copyrighted by the ACL up until December 2015. After removing some noisy examples such as papers with the keywords "proceedings" or "keynote" in them, as well as abstracts with email addresses in them, we were left with **16,845** training examples, **500** dev set examples and **500** test examples. This is an extremely small dataset by deep learning standards and poses a fair number of challenges as described when analyzing the results.

In this study, we restrict the abstract to the first three sentences. In Figure 2, we see the unigram overlaps between the title and abstract. As we keep more sentences in the abstract the fraction of overlap grows, however, so does the computational complexity and the amount of information that the encoder final representation needs to capture/attend to. Hence, we choose to use the first 3 sentences as an ideal point with 30% overlap. In addition, we have provided some additional statistics for the dataset in Table 1.

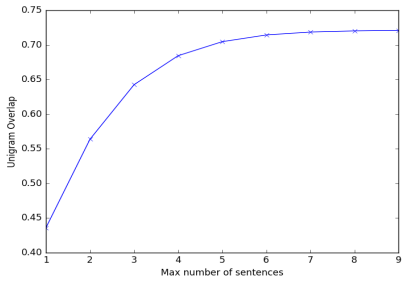


Figure 2: Unigram Overlap between title and abstract as a fraction of the shorter sequence (training set).

Statistic	Source	Target
Vocab Size	32,716	10,280
Max. Seq Length	929	52
Avg. Seq Length	125.1	9.7
Max. Seq Length (first 3 sents)	264	-
Avg. Seq Length (first 3 sents)	74.6	-

Table 1: ACL dataset statistics

5.2 Implementation Details

The model has been written entirely by the authors using Lasagne[1], a deep learning library built on top of Theano. Below we provide some training and evaluation setting details.

- Since the dataset is small, we use a shallow LSTM network with a single layer and 512 hidden units. All biases are initialized to 0. All weights are initialized using the Glorot Uniform initializer.
- We use an embedding layer of dimensionality 512 on top of the inputs which are sampled uniformly at random from $Unif[-.1, .1]$. The dimensionality matches the number of hidden units to reduce the number of parameters that need tuning.
- We use gradient clipping because the LSTM does not alleviate the exploding gradients issue, which the model encounters.
- Each source sequence length is limited to 250 tokens and each target sequence length is restricted to 25 tokens. Each target sequence is prepended with a special <EOS> token.
- At training, we try three different training schemes: (1) always decoding with the previous gold token, (2) data-as-demonstrator as described in Section 4, which is the same as the greedy decoding used at evaluation, (3) flipping a coin to decide between using the previous gold token and the previous timestep's output.
- At evaluation, we always use greedy decoding, i.e. use the previous timestep's output token $\hat{y}_{t-1}$, as the current timestep's input. We generate output tokens until the special <EOS> token is seen.

5.3 Hyperparameter Optimization

The first order of things when training the model was to overfit a tiny subset of the data. Figure 3 shows the training curves for numerous hyperparameter settings. We had two hyperparameters to tune, the learning rate and the gradient clipping threshold. In the figure we report log perplexity, but since $\log$ is a monotonic function, this does not alter the optimization problem.

We chose to use a learning rate of 0.3 and a gradient clipping threshold of 5 since the curve was relatively more stable and also converged quickly. The model was able to perfectly predict the titles of this tiny subset. The spikes on the various settings all indicate the model passing over a highly non-convex region of the space causing sharp spikes in training perplexity. But, as we can see, the model recovers from those spikes almost immediately.

5.4 Experimental results and Sample output

Training with the simplest model we see that the model can overfit the training data reasonably well. The log perplexity for "Vanilla seq2seq" (using gold tokens at training) in Figure 4 is below 2, which

[1] http://lasagne.readthedocs.io/en/latest/

Figure 3: Overfitting to a tiny training set.

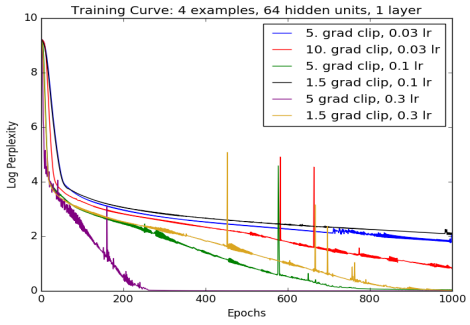
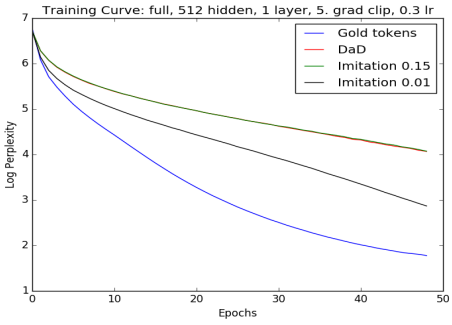


Figure 4: Three different training schemes.



means that perplexity is close to 7. Yet, when we decode training examples using this model that has seemingly overfit them, we find that it is unable to. Some sample decodings from the training set are shown in Table 2. As we can see in Figure 5, the representations computed for a sampled subset of the training data are identical. (This was repeated several times and it was identical each time.) This indicates that the encoder does not learn anything, which means the language model i.e. the decoder is able to learn/memorize the dataset perfectly, explaining the low perplexity.

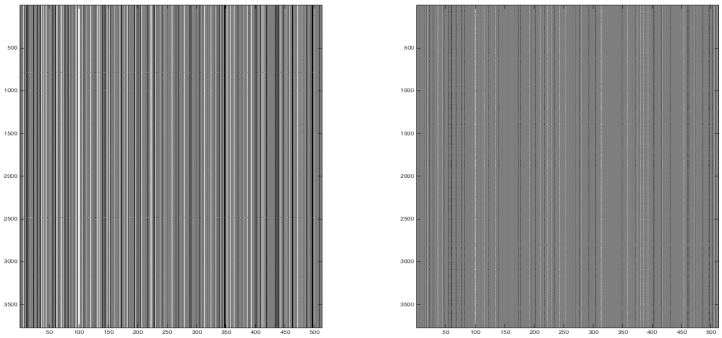


Figure 5: Visualizing the final hidden and cell states computed by the encoder.

Given the encoder representations, it can be seen that even though the model is overfitting, traditional regularization will not solve the issue because the model is unable to make predictions on the training data that it is overfitting. Hence, we need to try to force the encoder to learn different representations for the different examples. For this reason, we attempt to make training harder for the model, which would require the encoder to learn distinguishing representations for each example.

In Figure 4, we have curves for "DaD", which is greedy decoding test time, as well as "Imitation", which is the imitation learning technique - greedy decoding given coin flips - described in Section 4. 0.15 and 0.01 indicate that we use model output 15% and 1% of the time respectively. As can be seen from the figure, adding in chagnes to the training regime raised the training perplexity. The predictions and the hidden and cell representations, however, did not show any improvement. We can also see from the sample predictions in Table 2 that the general behavior of the model and its reliance on the language model did not change. One thing to note, however, is that the addition of imitation learning behaved as a regularizer for the language model, making it harder for it to memorize the training data. In fact, predictions for DaD and imitation learning are far longer than the vanilla seq2seq model. However, this still did not force the encoder to learn anything more.

5.5 Other attempts

There were a few pther things that we tried with the model to try to push the encoder to learn. Since these are predominantly bells and whistles that usually help a model that achieves some baseline performance, we omit the details in this paper. One of the first things we tried was a soft attention

Vanilla seq2seq
Pred: learning lexical semantic representations with web - scale knowledge bases True: machine translation evaluation with entailment
Pred: task : detecting semantic textual similarity using relational similarity measures True: the independence of dimensions in multidimensional dialogue act annotation
Pred: task : detecting semantic textual similarity using relational similarity measures True: the university of cambridge russian - english system at wmt13
DaD
Pred: translation evaluation using joint learning to dependency parsing and pos tagging with global features True: machine translation evaluation with entailment
Pred: automatic acquisition of semantic class acquisition from a simple approach to semantic role labeling with application to machine learning to semantic role labeling with maximum True: the independence of dimensions in multidimensional dialogue act annotation
Pred: automatic acquisition of semantic class acquisition from a simple approach to semantic role labeling with application to machine learning to semantic role labeling with maximum True: the university of cambridge russian - english system at wmt13
Imitation Learning
Pred: translation using source - source discriminative training for relation extraction in community question answering True: machine translation evaluation with entailment
Pred: automatic extraction of domain - specific sentiment in in in in in parsing systems True: the independence of dimensions in multidimensional dialogue act annotation
Pred: automatic extraction of domain - specific sentiment in in in in in parsing systems True: the university of cambridge russian - english system at wmt13

Table 2: Sample Predictions.

model to see if the encoder would learn which parts of the abstract to attend to, but it did not learn a better representation and the model continued to rely on solely the language model.

We also tried data augmentation to alleviate the small dataset issue. We did this by creating multiple copies of the same example, with the most frequent words dropped. The hope here was that variation in the source would help the encoder distinguish between examples at least a little. But the mapping from source to target seemed to be completely irrelevant to the model. In fact, since it is not possible to add more variation to the target sentences, even a 10x increase in the dataset size did not help and the decoder continued to memorize the examples to achieve a fairly low perplexity.

6 Conclusion and Future Work

In this paper, we attempted to build a neural network based text summarization model that could operate on a single domain scientific publication corpus. We found that a simple sequence to sequence model was prone to memorizing the target space in the training set, but was unable to learn anything useful enough to help it to generalize. In order to make it harder for the model to memorize the space and to force it to rely on the source, we modified training to use imitation learning where the model has to make a decision between using the model generated output vs. the gold token at each timestep. The expectation was that this would make minimizing the loss very hard and would push the model to use its source representation. However, instead of utilizing the source, the model simply performed worse.

Hence, learning a neural summarization model on a dataset of limited size is a hard task. In addition, this problem of predicting paper titles from abstracts in a abstractive summarization setting is new and no benchmarks exist to suggest that this problem is of equal difficulty as predicting news article titles from their first paragraph. We would like to explore this task further by using a multi-domain dataset of larger size. While this would increase variation in source to target mappings, it would allow the model to learn more than just a language model since the output space would be too large.

Once it can be confirmed that the task is not much harder than the one tackled by Rush et. al., we can try to understand how much data is necessary for the model to learn more. Text summarization for scientific publications is an incredibly important task, given the vast amount of literature available on each topic. Hence, understanding what makes it difficult and building robust models to solve it are important open problems.

References

Aone, C., Okurowski, M. E., Gorlinsky, J., and Larsen, B. (1999). A trainable summarizer with knowledge acquired from robust nlp techniques. *In Mani, I. and Maybury, M. T., editors, Advances in Automatic Text Summarization*, pages 7180. MIT Press.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, Montreal, Canada.

Bird, S., Dale, R., Dorr, B., Gibson, B., Joseph, M., Kan, M., Lee, D., Powley, B., Radev, D., and Tan, Y. F. (2008). The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In *Proc. of Language Resources and Evaluation Conference (LREC 08)*, Marrakesh, Morocco.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Graves, A. (2013). Generating sequences with recurrent neural networks. In *CoRR* arXiv:1308.0850.

Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In *Proceedings SIGIR 95*, pages 6873, New York, NY, USA.

Nallapati, R., Zhou, B., Santos, C. D., Gulcehre, C., and Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *CoRR*, arXiv:1602.06023.

Osborne, M. (2002). Using maximum entropy for sentence extraction. In *Proceedings of the ACL02 Workshop on Automatic Summarization*, pages 18, Morristown, NJ, USA.

Rush, A., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379-389, Lisbon, Portugal.

Sutskever, I., Vinyals, O., and Le, Q.V. (2014). Sequence to Sequence Learning with Neural networks. In *Advances in Neural Information Processing (NIPS)*, Montreal, Canada.

Venkatraman, A., Hebert, M., and Bagnell, J. A. (2015). Improving Multi-step Prediction of Learned Time Series Models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, Austin, TX, USA.