Document Summarization (draft)

Jeffrey Ling

February 23, 2017

Abstract

We propose a novel coarse-to-fine attention method to modify the standard sequence-to-sequence (seq2seq) attention model. By organizing the source sequence into a 2-dimensional image, we hierarchically apply attention, using a coarse stochastic mechanism for the first layer to select a row sequence for the second soft attention layer. While the computation for training standard seq2seq models scales linearly with source sequence length, our method is invariant to length and thus can scale arbitrarily.

We evaluate our model on the CNN/Dailymail document summarization task. As of this report we are able to match the performance of soft attention baselines using our new hierarchical method.

Contents

1 Introduction **2**
- 1.1 Natural Language Processing . . . 2
- 1.2 Deep Learning . . . 2
- 1.3 Our problem . . . 2
  - 1.3.1 Motivation . . . 3

2 Related Work **4**
- 2.1 Summarization . . . 4
- 2.2 Methods . . . 4
  - 2.2.1 Classical . . . 4
  - 2.2.2 Deep Learning . . . 5
  - 2.2.3 In the wild . . . 5
- 2.3 Neural Architectures . . . 5
- 2.4 Reinforcement Learning . . . 5

3 Models **7**
- 3.1 Sequence-to-sequence (seq2seq) . . . 7
  - 3.1.1 Recurrent encoder and decoder . . . 7
  - 3.1.2 Model 0: Standard Attention . . . 7
  - 3.1.3 Model 1 and 2: Hierarchical seq2seq . . . 8
- 3.2 Model 3: Hard Attention . . . 8
  - 3.2.1 Multiple Samples . . . 9
- 3.3 Reinforcement Learning . . . 9
  - 3.3.1 Variance Reduction . . . 9
  - 3.3.2 Curriculum . . . 10

4 Experiments **11**
- 4.1 Data . . . 11
  - 4.1.1 Yuntian's stuff . . . 11
  - 4.1.2 CNN/Dailymail . . . 11
- 4.2 Synthetic Pretraining . . . 12
- 4.3 Implementation Details . . . 12
- 4.4 Models . . . 12
  - 4.4.1 Baselines . . . 12
  - 4.4.2 Our models . . . 12

4.5 Training 12

5 Results **14**
5.1 Evaluation 14
5.2 Analysis 14

6 Discussion **15**

7 Conclusion **16**

Chapter 1

Introduction

1.1 Natural Language Processing

Natural language processing is a field with a long history.

The first methods were rule-based.

Previous methods were mainly statistical. The main algorithms for problems such as machine translation relied on collecting counts over large corpuses of data (**?**). Google Translate has relied on such methods to build state-of-the-art systems.

However, we are now in the age of deep learning. For the first time, neural networks effectively learning structure and features from language almost completely from scratch. The success of neural methods has been adopted by Google Translate to build even better systems Wu et al. (2016). These methods have attracted attention from the press

1.2 Deep Learning

Krizhevsky et al. (2012) won the ImageNet competition in ASDF year, beating the competition by a significant margin. This led to a new wave of research in neural networks.

Zeiler and Fergus (2014) show that deep networks such as the AlexNet can learn intermediate representations.

Mikolov and Dean (2013) show that training deep networks can also work on language.

Deep learning methods have since come to bear in NLP. Goldberg (2015) provides a nice summary.

Deeper and deeper networks become more and more feasible as modern computing power increases. Today, GPUs are leveraged to rapidly train networks at 10 or 20 times the speed of standard CPUs.

1.3 Our problem

Text summarization is an important problem for compressing large bodies of natural text into a more easily digestable form. Document summarization is one of the most challenging formulations of this problem, where given a document with several sentences of text, the goal is to produce a coherent summary that captures most or all of its salient points. To accomplish this, we use the

most recent advances in deep learning to automatically produce summaries by iterating over a large dataset of given examples.

We can frame summarization as a sequence to sequence task: given a news article (the source sequence of words), we desire to produce a summary (the target sequence). Existing methods in deep learning have been developed and proven to be highly effective for this kind of task, especially the sequence-to-sequence (seq2seq) model applied to machine translation (Sutskever et al., 2014; Dzmitry Bahdana et al., 2014).

Existing seq2seq methods are limited by the length of source and target sequences; for a problem such as document summary, the source sequence is too long for seq2seq models to be effective (in both a computational sense and correctness sense). In order to scale these methods for this application, we are experimenting with reinforcement learning methods that aim to prune down the length of the source sequence in an intelligent way.

1.3.1 Motivation

Why deep models? There is a debate in the deep learning community about the extent to which human inductive biases ought to be incorporated into deep networks. On one hand, machine learning systems often incorporate handcrafted features or assumptions in the model to solve the problem. Examples

On the other hand, a purely data-driven system forgoes these details, using highly general models for the task at hand.

In the past, NLP was done with handcrafted features. However, Collobert et al. (2011) showed that this is an unnecessary assumption:

There are ??? reasons why deep networks are desirable. First, they allow us to more easily automate the process of machine learning – by removing the human in the loop, we can develop more flexible models that save us from hard-coding in assumptions about the task at hand.

Second, removing assumptions about the input to a system allows us to build more general models – this has already proven to be useful in machine translation, where one system can be applied to many different settings. If we used a system of parsers

The vast majority of work so far in summarization has relied on simple, linguistically-motivated methods – ILP solvers, parsers / coref / asdfasdfasdfasdfasdf

It is not yet clear if a purely neural method can achieve the same success. This is the question we set out to resolve.

Chapter 2

Related Work

2.1 Summarization

Originated with Luhn (1958)

Summarization has historically been a difficult problem. To standardize the task, NIST released data for DUC (Document Understanding Conferences) between 2001-2007 (Over et al., 2007). The DUC tasks involved producing summaries for both single- and multiple-document sets of news articles. DUC 2001 and 2002 ask for general summaries of these articles documents and summaries, while DUC 2003-2006 also evaluate summaries based on their usefulness for certain question-answer tasks. While a single most effective metric for summarization may not exist, the DUC conferences established several important criteria, including grammaticality, non-redundancy, and content coverage (for which metrics like ROUGE (Lin, 2004) were created).

2.2 Methods

2.2.1 Classical

A variety of approaches have been used to solve summarization. Some of the most popular methods have been *extractive*, where sentences and words are pulled from documents based on relevance and pasted together (usually in document order). Some examples are Carbonell and Goldstein (1998), which uses a simple information metric for ranking sentences, and Svore et al. (2007), which uses a simple neural network for the same purpose.

Extractive: CRF Shen et al. (2004)

Another approach has been to relax general document summarization into the easier problem of sentence compression, where the source document consists of a single source. Knight and Marcu (2002) employs a noisy channel model, similar to machine translation, to deduce the "most probable" compression, while Clarke and Lapata (2008) uses an integer linear program. Cohn and Lapata (2008) extend the tree-based methods to allow for insertions and substitutions during compression, whereas prior methods were purely deletion based. Zajic et al. (2004) successfully use a sentence compression algorithm along with an unsupervised topic model on the DUC 2004 task.

While extraction has proven to be successful, the method is inherently limited in its ability to summarize. The more challenging method, and also the closest to what humans do, is *abstractive*

summarization. Instead of strictly requiring that all words of the summary come from the source document, any coherent text is allowed. In the past, such methods were limited to sentence compression tasks, as capturing inter-sentence dependencies and using a general vocabulary were beyond the scope of most statistical methods.

2.2.2 Deep Learning

With the onset of deep learning and sequence-to-sequence models, the abstractive approach has been reconsidered, as the only bottleneck for these methods is the quantity of data. Rush et al. (2015) proposed a data-driven, completely abstractive model for summarizing short sentences by training a seq2seq model with attention. More recent work (Nallapati et al., 2016; Ramachandran et al., 2016) scale the models to full documents, demonstrating that such methods are feasible.

These new models require a large amount of supervised training data, which now exist thanks to annotated CNN and Dailymail news stories (DUC data are limited in size and only suitable for evaluation). The task has not yet been fully standardized in this context, and researchers continue to debate what datasets are useful and what evaluation metrics to use (Toutanova et al., 2016).

2.2.3 In the wild

Reddit bot: `smmry.com`. Extractive, uses a ranking of sentences based on words (perhaps tf-idf?)

2.3 Neural Architectures

Sequence-to-sequence methods have been highly successful in many tasks, including machine translation (Sutskever et al., 2014; Dzmitry Bahdana et al., 2014), question answering (Hermann et al., 2015), dialogue (Li et al., 2016a), caption generation (Xu et al., 1994), and in particular summarization (Rush et al., 2015).

Li et al. (2015) envisions a hierarchical LSTM system, where a neural autoencoder combines sentence representations into a paragraph representation. The idea of separating text into levels of abstraction is also used in various forms of "hierarchical attention". Nallapati et al. (2016) applies the idea to summarization,

The problem of full document summarization, however, is still very open. In order to make the models work, Nallapati et al. (2016) use a variety of tricks: limiting the decoder vocabulary to the document, the use of pointer attention for `<unk>` tokens, etc. Our goal in this paper is not necessarily to optimize performance, but to understand how to scale up existing seq2seq methods in an efficient way, and so we attempt to eliminate the use of these ad-hoc tricks whenever possible. We aim to find the best implementation of "hard" attention, in which a discrete subset of the source document is selected at any given time, to satisfy the scalability condition.

Many techniques have been recently invented to handle the problem of scale. Martins and Astudillo (2016) propose the sparsemax function as a sparse alternative to softmax, and Rae et al. (2016) use a nearest neighbors approach in "sparse access memory" to train a large-scale neural Turing machine.

We will discuss one approach to sparseness in detail: reinforcement learning.

2.4 Reinforcement Learning

These methods are designed to be fully differentiable and hence compatible with standard backpropagation. Reinforcement learning, however, is an alternative method for obtaining "hard" attention. In traditional reinforcement learning, an agent is trained to maximize the total expected reward for performing a series of actions. Such methods can be applied to deep neural networks by backpropagating an unbiased gradient at the action step, which is commonly known as the REINFORCE algorithm or policy gradient (Williams, 1992; Schulman et al., 2015).

Deep reinforcement learning has been tried in the context of NLP (Zaremba and Sutskever, 2015; Ranzato et al., 2016; Li et al., 2016b) with varying degrees of success so far. We experiment with this in our paper.

Inspired by these ideas, we aim to improve on previous work for document summarization by (1) strengthening the hierarchical assumption with coarse features at the sentence level, and (2) including reinforcement learning, or one of recent sparse attention methods, for sparse attention.

In the next chapter we describe our models in detail.

Chapter 3

Models

3.1 Sequence-to-sequence (seq2seq)

We first describe the neural network architecture of the seq2seq models, also known as encoder-decoder models.

3.1.1 Recurrent encoder and decoder

As described in Dzmitry Bahdana et al. (2014), an *encoder* recurrent neural network (RNN) reads the source sequence as input to produce a vector known as the *context*, and a *decoder* RNN generates the output sequence using the context as input. One popular RNN choice is the long-short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997).

More formally, a given sentence $w_1, \ldots, w_n \in \mathcal{V}$ is transformed into a sequence of vectors $x_1, \ldots, x_n \in \mathbb{R}^{d_{in}}$ through a word embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d_{in}}$ as $x_t = Ew_t$. An RNN is given by a parametrizable function f and a hidden state $h_t \in \mathbb{R}^{d_{hid}}$ at each time step t with $h_t = f(x_t, h_{t-1})$. This sequence of hidden states $h_1, \ldots, h_n$ jointly forms the context $c_{enc} = [h_1, \ldots, h_n]$ that is passed to the decoder.

The decoder is another RNN f_{dec} that generates output words $y_t \in \mathcal{V}$. It keeps hidden state $h_t^{dec} \in \mathbb{R}^{d_{hid}}$ as $h_t^{dec} = f_{dec}(y_{t-1}, h_{t-1}^{dec})$. Each word is predicted using another function g as

$$p(y_t | y_{t-1}, \ldots, y_1, c) = g(h_t^{dec}, c_{enc})$$

The models are trained to maximize the log probability of getting the sequences in the dataset correct. As the model is fully differentiable with respect to its parameters, we can train it end-to-end with stochastic gradient descent and the backpropagation algorithm.

The details of the function g will be described next.

3.1.2 Model 0: Standard Attention

In Dzmitry Bahdana et al. (2014), the function g is implemented with an *attention network*. We compute attention weights for each encoder hidden state h_i as follows:

$$\beta_i = h_i^T W h_t^{dec}$$

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{j=1}^{n} \exp(\beta_j)}$$

$$\widetilde{c} = \sum_{i=1}^{n} \alpha_i h_i$$

We normalize the α_i to sum to 1 over the source sentence words.

g is then implemented as

$$g(h_t^{dec}, c) = W_{out}(\widetilde{c}^T W_2 h_t^{dec}) + b_{out}$$

In essence, g computes a probability distribution α over the encoder hidden states, then takes the expectation of the encoder hidden state under α. The idea behind attention is to select the most relevant words of the source (by assigning higher attention weights) when generating output word y_t at time t.

Going forward, we call this *Model 0*.

3.1.3 Model 1 and 2: Hierarchical seq2seq

For a large source input like a document, it may be computationally inefficient to run an RNN over the entire source. Instead, we can consider organizing the document into distinct sentences and run an RNN separately over each. Specifically, assuming we have sentences $s_1, \ldots, s_m$ with words $w_{i,1}, \ldots, w_{i,n_i}$ for sentence s_i, we can apply an RNN to get corresponding hidden states $h_{i,j}$.

For attention, we then have two options. We can follow Model 0 and compute attention weights $\alpha_{i,j}$ for each hidden state $h_{i,j}$ by normalizing over all states. We call this *Model 1*.

Alternatively, rather than taking attention over the entire document, we can instead have a two-layered hierarchical attention mechanism: first, we have weights $\alpha_1, \ldots, \alpha_m$ for each sentence, and then for each sentence s_i, we have another set of weights $\widetilde{\alpha}_{i,1}, \ldots, \widetilde{\alpha}_{i,n_i}$.

The sentence level attention is computed using a different method: we first produce a representation of each sentence given the words $x_{i,1}, \ldots, x_{i,n_i}$ of the sentence. Our first option is *bag of words*: we simply take the sentence representation $u_i = \sum_{j=1}^{n_i} E x_{i,j} \in \mathbb{R}^{d_{in}}$, i.e. the sum of the word embeddings.

Alternatively, we can use a convolutional method: as in Kim (2014), we perform a convolution over each window of words in the sentence. We use max-over-time pooling to obtain a fixed-dimensional sentence representation in $\mathbb{R}^{d_f}$ where d_f is the number of filters.

The final attention computed for word j in sentence i will thus be

$$\alpha_{i,j} = \alpha_i \cdot \widetilde{\alpha}_{i,j}$$

following the interpretation of $\alpha_{i,j}$ as the probability mass on $w_{i,j}$.

We call this method of attention *Model 2*.

3.2 Model 3: Hard Attention

With hierarchical attention, we still do not obtain significant gains in efficiency, since we have to compute RNN states for all words in the source document. Therefore, the idea that underlies this project is to apply stochastic sampling methods to the attention distribution α.

Specifically, rather than computing the context $\widetilde{c} = \sum_i \alpha_i h_i$, we can sample from the probability distribution α_i to obtain a single state h_i, and we set $\widetilde{c} = h_i$ as the sampled hidden state.

Known in the literature as "hard attention" (Xu et al., 1994), this model loses the property of being end-to-end differentiable and thus cannot be trained with standard backpropagation. However, reinforcement learning provides a way to circumvent this issue, as described below.

We take Model 2 and apply hard attention at the sentence level, but keep the word level attention per sentence as is. That is, we sample from the attention weights $\alpha_1, \ldots, \alpha_m$ to obtain a one-hot encoding for the sentence attention, and apply the same multiplication with this one-hot vector on the word-level attention weights $\widetilde{\alpha}_{i,1}, \ldots, \widetilde{\alpha}_{i,n_i}$. We call this *Model 3*.

3.2.1 Multiple Samples

From our initial experiments with Model 3, we found that taking a single sample was not very effective. However, we discovered that sampling multiple times from the distribution α significantly improves performance.

We sample based on the multinomial distribution $\text{Mult}(k, \{\alpha_i\}_{i=1}^n)$ to produce the sentence-level attention vector α of length n, with $\alpha_i = x_i/k$, where x_i is the number of times index i was sampled. k is a hyperparameter which can be tuned, and we found that $k = 5$ works well in our experiments.

The intuition here is for the hard attention model to more closely approximate the soft attention model, as it can select more sentences to produce the context.

3.3 Reinforcement Learning

To train Model 3, we must apply techniques from the reinforcement learning literature. We cast our learning problem with the neural network as stochastic agent and log probability as reward, and thus are able to apply reinforcement learning to train the network.

In our setup, where our agent is a parameterized model, computing the gradients for the model in this setup is known as the REINFORCE algorithm (Williams, 1992) or policy gradient, and has been well-explained in recent work (Mnih et al., 2014; Ba et al., 2015; Schulman et al., 2015).

Specifically, assuming we have a probability distribution $p(\alpha)$ from which we sample, and subsequently receive reward r, the gradient that is backpropagated from $p(\alpha)$ is

$$\nabla_\theta \mathcal{L}(\theta) = r \nabla_\theta \log p(\alpha) \tag{3.1}$$

i.e. the reward multiplied by the gradient of the log probability. A proof of this can be found in Williams (1992).

In our framework, we use the log probability of the correct word at each time step as the reward r_t. Since samples at time t of the RNN decoder can also affect future rewards, we use a discount factor of $\gamma = 0.5$, so that the reward is $r = \sum_{s=t}^n \gamma^{n-s} r_s$ for the decoding sampler at time t.

3.3.1 Variance Reduction

While the policy gradient of equation 3.1 is proven to be unbiased, in practice it has such high variance that training converges far too slowly.

One of the most common ways to reduce the variance of the gradient estimator is to introduce a baseline reward b, which we subtract from our reward. Including a baseline is proven to reduce the variance of the estimator (Mnih and Gregor, 2014). We also normalize the rewards to a common scale by dividing by the reward variance in a given minibatch.

Our policy gradient equation then becomes

$$\nabla_\theta \mathcal{L}(\theta) = \frac{r - b}{\sigma} \nabla_\theta \log p(\alpha) \tag{3.2}$$

There are a few methods for producing the baseline; as in Mnih and Gregor (2014), we can keep an exponentially moving average of the reward

$$b_j = (1 - \beta) b_{j-1} + \beta r_j$$

where r_j is the average minibatch reward and β is a hyperparameter (set to 0.9). Similarly, we keep a moving average of the variance for normalization:

$$\sigma_j^2 = (1 - \beta) \sigma_{j-1}^2 + \beta v_j$$

where v_t is the variance of the minibatch rewards for batch j. Since we have rewards at each time step of the decoder LSTM, we keep a separate moving average for the baseline for each time step, but we keep a single moving variance for all time steps.

While several papers suggest using a learned baseline from the RNN state (e.g. Ranzato et al. (2016)), we have not found this to be more effective.

We also use an entropy term to reduce the variance. Our policy gradient equation then becomes

$$\nabla_\theta \mathcal{L}(\theta) = \frac{r - b}{\sigma} \nabla_\theta \log p(\alpha) - \lambda_{ent} \nabla_\alpha (\alpha \log \alpha) \tag{3.3}$$

where λ_{ent} is a hyperparameter. This has the effect of increasing the entropy of our sampling distribution, hence encouraging more exploration and faster convergence of learning.

3.3.2 Curriculum

Since training using policy gradients tends to be noisy and slow to converge, we experimented with a curriculum that starts training with soft attention and in epoch t, trains a minibatch using hard attention with probability $p_t = 1 - 1/\sqrt{t}$ (Gülçehre et al., 2016).

While we found this to be helpful for single sample hard attention, it was not necessary for effective training with multisampled hard attention. We prefer to train solely with hard attention when possible, as we are able to save computation at training time.

Chapter 4

Experiments

4.1 Data

4.1.1 Yuntian's stuff

Table ???? for Yuntian

4.1.2 CNN/Dailymail

Experiments were performed on the CNN/Dailymail dataset from Hermann et al. (2015). While the dataset was created for a question-answering task, the dataset format is suited for summary. Each data point is a news document accompanied by up to 4 "highlights", and we take the first of these as our target summary. Train, validation, and test splits are provided along with document tokenization and sentence splitting. We do additional preprocessing by replacing all numbers with # and appending end of sentence tokens to each sentence. We limit our vocabulary size to 50000 most frequent words, replacing the rest with `<unk>` tokens. We dropped the documents which had an empty source (which came from photo articles).

Table 4.1 lists statistics for the CNN/Dailymail dataset.

Dataset	CNN	Dailymail
Train size	90267	196962
Valid size	1221	12149
Average words per doc	794	832
Average sents per doc	21	29
Average words per sent	36	27
Average words per summary	13	15

Table 4.1: Statistics for CNN/Dailymail data.

4.2 Synthetic Pretraining

We found that unsupervised pretraining on the given dataset is beneficial to learning. For each document, we randomly sample 2 sentences and concatenate to form the target sentence in a new synthetic dataset. We can sample multiple times to have multiple targets for a given source document, and we found that 5 samples was most beneficial to learning (performance drops with significantly more samples). We then train on the synthetic dataset for 5 epochs and initialize future training with the learned weights.

4.3 Implementation Details

A few implementation details were necessary to make minibatch training possible. First, instead of taking attention over each individual sentence, we arrange the first 400 words of the document into a 10 by 40 image, and take each row to be a sentence. Second, we pad short documents to the maximum length with a special padding word, and allow the model to attend to it. However, we zero out word embeddings for the padding states and also zero out their corresponding LSTM states. We found in practice that very little of the attention ended up on the padding words.

Ideally, we would prefer to not truncate documents, especially since later context can be important for summarizing the document. Due to memory issues, this is a problem we still have to resolve.

4.4 Models

4.4.1 Baselines

For a baseline, we take the first 15 words of the document (chosen as the average length of a sentence in the training dataset). We call this FIRST.

4.4.2 Our models

We ran experiments with Models 0 to 3 as described above. Model 0 serves as the baseline.

- Model 0: Soft attention.
- Model 1: Hierarchical LSTM, soft attention over all.
- Model 2: Hierarchical LSTM, soft hierarchical attention.
- Model 3: Hierarchical LSTM, hard attention over sentences.
- Model 3+multisampling: We include multisampling with $k = 5$.

4.5 Training

We train with minibatch stochastic gradient descent (SGD) with batch size 32 for 13 epochs, renormalizing gradients to be below norm 5. We initialize the learning rate to 1, and begin decaying it by 0.5 each epoch after the validation perplexity stops decreasing.

We use 2 layer LSTMs with 500 hidden units, and we initialize word embeddings with 300-dimensional word2vec embeddings (Mikolov and Dean, 2013). For convolutional layers, we use a kernel width of 6 and 600 filters.

In the next chapter we show results.

Chapter 5

Results

5.1 Evaluation

We report metrics for best validation perplexity and ROUGE scores (Lin, 2004). We use the ROUGE balanced F-scores with ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common substring). We chose F-scores since recall is biased towards longer sentences.

To generate summaries for evaluation, we run beam search with a beam size of 5.

Model	PPL	ROUGE-1	ROUGE-2	ROUGE-L
FIRST	-	23.1	9.8	20.5
BERKELEY				
MODEL 0				
MODEL 1				
MODEL 2	16.2	24.5	12.0	22.9
MODEL 2+SYNTHPRE	16.0	23.7	11.5	22.0
MODEL 3				
MODEL 3+SAMPLE5	17	23.9	11.3	22.4
MODEL 3+SAMPLE5+SYNTHPRE	14.6	24.1	11.7	22.5

Table 5.1: Summarization results for CNN/Dailymail.

5.2 Analysis

See Table 5.1 for summary results.

We notice that Model 2 has the best performance, while multisampling is comparable. We hypothesize that by sampling multiple times, the model learns to approximate the soft attention distribution

Chapter 6

Discussion

This method probably doesn't work.

Chapter 7

Conclusion

We tried a new neural architecture that failed.

Bibliography

Jimmy Lei Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition With Visual Attention. *Iclr*, abs/1412.7755:1–10, 2015. URL http://arxiv.org/abs/1412.7755.

J Carbonell and J Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM. ISBN 1581130155. doi: 10.1145/290941.291025. URL papers2://publication/uuid/1FA33AEC-2C9E-4149-B740-02A7C6C24B93.

James Clarke and Mirella Lapata. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res.*, 31:399–429, 2008.

Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *Proc. 22nd Int. Conf. Comput. Linguist. 1*, pages 137–144. Association for Computational Linguistics, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011. ISSN 0891-2017. doi: 10.1.1.231.4614.

Dzmitry Bahdana, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation By Jointly Learning To Align and Translate. *Iclr 2015*, pages 1–15, 2014. ISSN 0147-006X. doi: 10.1146/annurev.neuro.26.041002.131047. URL http://arxiv.org/abs/1409.0473v3.

Yoav Goldberg. A primer on neural network models for natural language processing. *arXiv Prepr. arXiv1510.00726*, 2015.

Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes. *CoRR*, abs/1607.00036, 2016. URL http://arxiv.org/abs/1607.00036.

KM Hermann, T Kocisky, and E Grefenstette. Teaching machines to read and comprehend. *Adv. Neural*, pages 1–9, 2015. URL http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process. (EMNLP 2014)*, pages 1746–1751, 2014. ISSN 10709908. doi: 10.1109/LSP.2014.2325781. URL http://emnlp2014.org/papers/pdf/EMNLP2014181.pdf.

Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107, 2002.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A Hierarchical Neural Autoencoder for Paragraphs and Documents. *CoRR*, abs/1506.01057, 2015. URL http://arxiv.org/abs/1506.01057.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A Persona-Based Neural Conversation Model. *arXiv Prepr. arXiv1603.06155*, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. Deep Reinforcement Learning for Dialogue Generation. *arXiv*, 2(2):1192–1202, 2016b. URL http://arxiv.org/abs/1606.01541.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summ. branches out Proc. ACL-04 Work.*, volume 8. Barcelona, Spain, 2004.

Hans Peter Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, 1958.

André F. T. Martins and Ramón Fernandez Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. *Proc. 33rd Int. Conf. Mach. Learn.*, abs/1602.02068:1614–1623, 2016. URL http://arxiv.org/abs/1602.02068.

T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.*, 2013.

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. *ArXiv stat.ML*, 32(October):1–20, 2014. URL http://arxiv.org/abs/1402.0030.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *CoRR*, abs/1406.6247, 2014. URL http://arxiv.org/abs/1406.6247.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Proc. CoNLL*, abs/1602.06023:280–290, 2016. URL http://arxiv.org/abs/1602.06023.

Paul Over, Hoa Dang, and Donna Harman. DUC in context. *Inf. Process. Manag.*, 43(6):1506–1520, 2007. ISSN 03064573. doi: 10.1016/j.ipm.2007.01.019.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Adv. Neural Inf. Process. Syst. 29*, pages 3621–3629. Curran Associates, Inc., 2016.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. Unsupervised Pretraining for Sequence to Sequence Learning. *ICLR'17 Submiss.*, abs/1611.0:1–20, 2016. URL http://arxiv.org/abs/1611.02683.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence Level Training with Recurrent Neural Networks. *Iclr*, abs/1511.0:1–15, 2016. doi: 10.1371/journal.pcbi.1005055. URL http://arxiv.org/abs/1511.06732.

Alexander M Rush, Sumit Chopra, and Jason Weston. A Neural Attention Model for Abstractive Sentence Summarization. *Proc. Conf. Empir. Methods Nat. Lang. Process.*, (September):379–389, 2015. ISSN 19909772. doi: 10.1162/153244303322533223. URL http://arxiv.org/abs/1509.00685.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs. *Nips*, abs/1506.05254:1–13, 2015. ISSN 10495258. URL http://arxiv.org/abs/1506.05254.

Dou Shen, Jian-tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document Summarization using Conditional Random Fields. *Science (80-.).*, 7:2862–2867, 2004. ISSN 10450823. URL http://scholar.google.com/scholar?hl=en{\&}btnG=Search{\&}q=intitle:Document+Summarization+using+Conditional+Random+Fields{\#}0.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Adv. Neural Inf. Process. Syst.*, pages 3104–3112, 2014.

Krysta Marie Svore, Lucy Vanderwende, and Christopher J C Burges. Enhancing Single-Document Summarization by Combining RankNet and Third-Party Sources. In *EMNLP-CoNLL*, number August 2016, pages 448–457, 2007.

Kristina Toutanova, Ke M Tran, and Saleema Amershi. A Dataset and Evaluation Metrics for Abstractive Compression of Sentences and Short Paragraphs. In *EMNLP*, nov 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, 1992.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints*, pages 1–23, 2016. URL http://arxiv.org/abs/1609.08144.

Kelvin Xu, Jimmy Lei Ba Ryan Kiros, Kyunghyun Cho Aaron Courville, and Ruslan Salakhutdinov Richard S. Zemel Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994. ISSN 19410093. doi: 10.1109/72.279181. URL http://arxiv.org/abs/1502.03044.

David Zajic, Bonnie Dorr, and Richard Schwartz. Bbn/umd at duc-2004: Topiary. In *Proc. HLT-NAACL 2004 Doc. Underst. Work. Bost.*, pages 112–119, 2004.

Wojciech Zaremba and Ilya Sutskever. Reinforcement Learning Neural Turing Machines. *CoRR*, abs/1505.00521, 2015. URL http://arxiv.org/abs/1505.00521.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10590-1_53. URL http://link.springer.com/10.1007/978-3-319-10590-1{_}53{\%}5Cnhttp://arxiv.org/abs/1311.2901{\%}5Cnpapers3://publication/uuid/44feb4b1-873a-4443-8baa-1730ecd16291.