Document Summarization (draft)

Jeffrey Ling

March 14, 2017

Abstract

Standard sequence-to-sequence (seq2seq) attention models have seen great success in NLP, but do not scale well to tasks with long sequences. We propose a novel coarse-to-fine attention method to reduce the number of computations necessary in standard attention. By organizing the source sequence into a 2-dimensional image, we hierarchically apply attention, using a coarse mechanism for the first layer to select a row sequence, and a finer mechanism 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.

Contents

Chapter 1

Introduction

1.1 Natural Language Processing

Natural language processing is a field with a variety of interesting structured prediction problems. The essential goal of NLP is to build a model of language so that computers can automatically process substantial quantities of text — a highly relevant problem in today's information age.

While humans have no trouble understanding and using language, even the simplest language tasks can be impossible for computers. Some classical NLP problems include part-of-speech tagging, parsing, language modeling, and machine translation.

It is informative to consider the history of machine translation. The first methods were rule-based linguistics systems, and the company SYSTRAN provided one of the first ().

In the 90s, statistical methods for NLP became increasingly popular. By considering language generation as a probabilistic process, one can collect statistics over large corpuses of data to automatically deduce the parameters of the model. In machine translation, this idea was used to great effect in Brown et al. (1993), whose count-based methods form the core of state-of-the-art systems like Google Translate.

Recently, researchers found that deep learning works effectively for many NLP tasks. For the first time, neural networks were able learning structure and features from language almost completely from scratch (Collobert et al., 2011). The success of neural methods has been adopted by Google Translate to build even better systems (Wu et al., 2016). Research in applying deep learning to NLP is ongoing.

1.2 Deep Learning

The history of neural networks dates back to the perceptron (Rosenblatt, 1958), a simple model that assumes data can be linearly separated. Due to this strict requirement, the machine learning community dismissed the idea as impractical, and research was sidelined for most of the 20th century.

Recently, neural networks have made a resurgence. In the ImageNet image classification competition in 2012, Krizhevsky et al. (2012) won using *convolutional neural networks* (LeCun and Bengio, 1995), beating the competition by a significant margin. This led to a renewed wave of research, especially due to the advancement of modern computing power and GPUs, which can train net-

works at 10 or 20 times the speed of standard CPUs. Today, deep models are used to successfully play Go (Silver et al., 2016), play Atari games from pixels (Mnih et al., 2015), and

While neural networks are often treated as black box classifiers, Zeiler and Fergus (2014) show that the intermediate layers of deep convolutional networks contain abstracted qualities of the input, such as patterns, textures, and objects of the input. This suggests that neural networks are discovering features of the input and building generalized *representations* of their inputs.

The idea of learning representations of the input is highly general, and so it comes as no surprise that deep networks soon found applications in NLP. Mikolov and Dean (2013) show that by training a neural network on a Google News text corpus, the network learned to map words in the English language to a vector of real numbers known as *word embeddings*. These word embeddings are actually able to capture semantic properties of the words — for example, taking the vectors for *king*, *man*, and *woman*, we find that $v_{king} - v_{man} + v_{woman} \approx v_{queen}$, preserving the analogy that we usually make with text.

Since the onset of deep learning, deep models have found their way into nearly every corner of NLP. Much of their success relies on the ubiquity of the *long short-term memory* (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997), a model used to both process and generate sequences of text. State-of-the-art systems for many problems, including machine translation, now use deep learning as their core algorithm.

1.3 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) shows that this is an unnecessary assumption. They build a neural network for classical NLP tasks (e.g. part-of-speech tagging, named entity recognition) that can learn to do the task without any feature extraction.

There are a few reasons why deep networks are desirable for NLP. First, they are remarkably powerful and practically easier to train than complicated rule-based systems (assuming enough computational power is available), and are not mutually exclusive with standard feature extraction methods. Second, they allow for non-domain experts to make progress on relevant problems. Third, we find that trained models can discover latent structure in language automatically, which is an interesting phenomenon that may reveal more about how language is used. As an example, sequence-to-sequence models with attention (Bahdanau et al., 2014) learn the concept of a word alignment in translation without any supervision.

It is not yet clear, however, to what extent neural methods can replace models with hard-coded linguistic assumptions. This is one of the questions we set out to resolve.

1.4 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 training on a large dataset of given examples.

We can frame summarization as a supervised 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; Bahdanau et al., 2014). Rush et al. (2015) uses the seq2seq model to summarize sentences into shorter headlines, but we consider the more general problem of arbitrarily long documents.

Existing seq2seq methods are limited by the length of source and target sequences. For a problem such as document summarization, the source sequence of length N requires $O(N)$ seq2seq model computations. However, it makes sense intuitively that not every word of the document will be necessary for generating a summary, and so we would like to reduce the amount of necessary computations over the source document.

Therefore, in order to scale seq2seq methods for this problem, we aim to prune down the length of the source sequence in an intelligent way. In the problem of image captioning, Xu et al. (2015) show that when generating words of the caption, the model places attention on the relevant part of the image. Inspired by this idea, we introduce the coarse-to-fine attention mechanism that sparsely selects subsets of the source document for processing.

We provide an outline for the rest of this thesis. In section ? we cover related work in summarization and deep learning. In section ? we give the necessary background for our models. In section ? we describe our models in detail. In section ? we show results and discussion.

Chapter 2

Related Work

2.1 Automatic Summarization

Nenkova and McKeown (2011) give an overview of the field. In particular, they provide a taxonomy of methods that researchers have used to frame summarization:

Extractive vs. Abstractive Extractive summaries extract certain sentences or phrases from the document, while abstractive summaries take a more holistic view and can in practice be anything (similar to how humans produce summaries).

Single- vs. Multi-document Summarization was originally posed as the problem of producing a summary for a single document. However, with the onset of the Internet and search engines, there are often multiple documents on the same topic, and so a summarization system should be able to use all of them to produce a summary.

Indicative (style) vs. informative (facts) Indicative summaries provide a sense for what a certain document is about without necessarily giving all of its details, while informative summaries are meant to replace the original document in terms of content.

Keyword (words) vs. headline (sentence) Keyword summaries are allowed to simply be a bag-of-words of important keywords from the document, while headline summaries must form a coherent sentence.

Generic vs. query focused Generic summaries have no assumptions on the reader and are meant to be generally informative, while query focused summaries take into consideration a query and only return relevant information. The contrast between these two methods highlights an important question in summarization: to what end are we summarizing documents? If we can answer this question more precisely, we will be better able to build systems to accomplish our desired tasks.

Before elaborating on specific methods, it is important to highlight these differences to understand how people thought about the problem. For example, extractive methods are by far more

popular due to the simplicity of building an extraction algorithm, while abstractive methods have been difficult as we do not have a fully working model of language generation.

Based on this taxonomy, the deep model we set out to build would be classified as 1) abstractive, 2) single-document, 3) informative, 4) headline, 5) generic.

In the rest of this thesis, we will limit the scope of the summarization problem to the *single-document*, *informative*, *headline*, and *generic* categories in order to focus on the application of deep learning in the problem.

In the next section, we give a brief overview of some of the relevant methods used in summarization.

2.2 Methods

2.2.1 Classical

One of the first considerations of automatically producing summaries was Luhn (1958), which aimed to summarize scientific articles using a sentence-ranking method. The algorithm gives each sentence a score based on the occurrence of frequently appearing words.

Since then, a variety of approaches have been used to solve summarization. We highlight some notable work in both the extractive and abstractive framework.

Extractive

The most popular methods for document summarization have generally been extractive due to their simplicity. One natural procedure for an extractive summarization is to score sentences based on some relevance metric and return the highest scoring sentences (perhaps reordering them as well).

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.

Shen et al. (2004) models sentence extraction as a sequential decision problem, using a linear-chain conditional random field to find the best subset of sentences.

Abstractive

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.

Two methods used to produce abstractive summaries are sentence compression and sentence fusion. Compression removes less useful information from sentences, while fusion is harder and combines information from sentences.

Compression: 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.

Fusion: align parse trees and combine phrases that are similar

2.2.2 Deep Learning

With the onset of deep learning, learning an end-to-end abstractive model for summarization has become more feasible. Rush et al. (2015) propose a data-driven, completely abstractive model for summarizing short sentences by training a sequence-to-sequence model with attention. More recent work in deep learning has been done for both extractive () and abstractive (Nallapati et al., 2016; Ramachandran et al., 2016) methods that scale the models to full documents, demonstrating the feasibility of end-to-end models.

These new models require a large amount of supervised training data, which the DUC data are unsuitable for due to their small scale. However, thanks to the large-scale annotated CNN/Dailymail news stories dataset Hermann et al. (2015), we now have the necessary data to train our deep models.

The task has not yet been fully standardized in this context, and research in the area is still largely preliminary. While CNN/Dailymail may not be the most suitable dataset for the task (Chen et al., 2016), a better alternative is yet to exist.

2.2.3 Datasets

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).

DUC overall was found not to have the best impact. For many of the news summary tasks, it was found that taking the first sentence of each article could not be beaten by more sophisticated methods, and so the task was somewhat abandoned.

CNN/Dailymail is much larger, and is the one we use to train our deep models.

2.2.4 Evaluation

Evaluating a good summary is inherently ambiguous, and probably one of the hardest parts of the problem.

For extractive summaries, people have proposed simple metrics such as precision and recall on selected sentences. These naturally do not work too well since 1) not all sentences are equally informative, and 2) not all parts of a sentence are relevant.

DUC really pushed forward understanding on evaluation. They came up with recall on elementary discourse units (EDUs), based on clauses within a summary that ought to be captured. ROUGE Lin (2004) is cheap and fast. Pyramid method is a complicated human evaluation method based on summary content units (SCUs).

None of these methods directly address the grammaticality of the output. Aside from using human evaluation, meaningful metrics for summaries is still very much an open question (Toutanova et al., 2016). In our work, we settle for ROUGE due to its cheapness and ease of use in evaluating our models.

2.2.5 In the wild

Summarization is an important real-world problem due to the explosion of available data. Thus, there are many practical methods that have been developed and deployed in real-world settings. One of the most notable examples is on Reddit[1]: in order to summarize long forum discussions, Reddit uses technology from SMMRY[2].

Smmry's algorithm is a simple extractive summarization method. It counts word occurrences, splits discussions by sentence, and ranks the sentences based on the sum of their word scores (perhaps tf-idf?). This algorithm bears extraordinary similarity to Luhn (1958) — although a variety of methods have been invented since then, the simplest approaches turn out to be the most practical.

[1] `reddit.com`
[2] `smmry.com`

Chapter 3

Background

In this chapter, we set up the relevant background ideas for our models.

3.1 Sequence-to-Sequence Attention Models

The sequence-to-sequence architecture (Sutskever et al., 2014), also known as the encoder-decoder architecture, forms the backbone of many successful models in NLP. A popular variant of sequence-to-sequence models are *attention* models (Bahdanau et al., 2014). The key idea is to keep an encoded representation of all parts of the input, attending to the relevant part each time we produce an output from the decoder. These models have been used to great effect in a variety of NLP tasks, including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), question answering (Hermann et al., 2015), dialogue (Li et al., 2016a), caption generation (Xu et al., 2015), and in particular summarization (Rush et al., 2015).

Xu et al. (2015) show how attention models can be used to "summarize" an image and produce a caption. By analyzing where in the image their models attend to when generating each word of the caption, they qualitatively find that the model is essentially describing that region of the image. Figure **??** shows some examples.

We can leverage the same idea for text summarization, assuming we have a suitable representation of our input document. The simplest method for doing so would be to run an LSTM over the document.

However, the attention step becomes computationally difficult — for each word we generate, we need to compare it to every word of the document in order to determine which part to attend to. Therefore, we propose a hierarchical method of attending to the document by first attending to sentences, then to the words within sentences. We call this method *coarse-to-fine attention* [1].

To be able to attend to both sentences and words in a hierarchical manner, we need to construct encodings of the document at both levels. Thus, we run a low-level LSTM encoder on the words of each sentence for a fine-grained representation of the text, and a simpler encoder model (e.g. bag of words) for coarse-grained sentence representations. Sukhbaatar et al. (2015) demonstrate how coarse representations can be useful by using memory networks to access information for simple

[1] The term coarse-to-fine attention has previously been introduced in the literature (Mei et al., 2016). However, their idea is different: they use coarse attention to reweight the fine attention computed over the entire input. Similar ideas have also been called hierarchical attention (Nallapati et al., 2016).

question-answering tasks. Li et al. (2015) use the idea of coarse and fine encodings to develop a hierarchical autoencoder for representing paragraphs of text.

Therefore, if we can make our model first use coarse attention to choose sentences, then use fine attention to choose words only from that sentence, then we avoid the computational cost of searching over the entire document. This idea runs into a very serious problem, however: by posing the attention as a discrete selection process, the neural network is no longer differentiable.

Xu et al. (2015) suggest "hard" attention as one possible solution to the discrete selection problem. While standard "soft" attention actually averages the representations of where the model attends to, for hard attention we make a hard decision and choose only one location. Such models can be trained using reinforcement learning. Before we elaborate on this method, we survey some other methods invented to overcome this discrete attention problem.

3.2 Conditional Computation

elaboration needed

Many techniques have been proposed in the literature to handle the problem of large inputs to deep neural networks.

The term "conditional computation" was coined by Bengio et al. (2013), where the idea is to compute a subset of units for a given network per input. This would have the advantage of being much more efficient, especially for networks that need to handle extremely large inputs as is common in vision and NLP.

While standard deep models use the softmax function $\text{softmax}(\boldsymbol{z})_i = \exp(z_i) / \sum_j \exp(z_j)$ to produce differentiable probability distributions, Martins and Astudillo (2016) propose the sparsemax function as a sparse alternative to softmax that still has a useful gradient.

Rae et al. (2016) use a nearest neighbors approach in "sparse access memory" to train a large-scale neural Turing machine.

Shazeer et al. (2017) introduces a mixture-of-experts model that selectively chooses some "expert" subnetworks in the spirit of conditional computation.

These methods are all promising and merit further research in the future. In this work, we limit our scope to the hard attention model of Xu et al. (2015). In the next section, we explain reinforcement learning and how it can be used to train the hard attention model.

3.3 Reinforcement Learning

Standard backpropagation training of neural networks assumes that the output is a differentiable function of the input. Reinforcement learning, however, is a general framework that makes no such assumptions.

The traditional setup of reinforcement learning assumes some agent is navigating an environment and earning rewards. We assume the agent is at state s_t in time t, makes an action a_t, earns a reward r_t, and transitions to the next state s_{t+1}. Assuming there is randomness in the reward and the transition $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, the agent wants to maximize total expected reward $\mathbb{E}_{s_t, r_t}[\sum_{t=0}^{\infty} \gamma^t r_t]$ where γ is a future discount factor.

It turns out that such a framework can be applied to deep neural networks to obtain gradients for backpropagation.

transition

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

$$\frac{\partial \mathcal{L}}{\partial w} = r \frac{\partial \log p(\alpha)}{\partial w} \tag{3.1}$$

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. (2015)), 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 Deep reinforcement learning

Deep reinforcement learning has been tried in the context of NLP (Zaremba and Sutskever, 2015; Ranzato et al., 2015; 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 for sparse attention.

In the next chapter we describe our models in detail.

Chapter 4

Models

In this chapter, we describe our models in full mathematical detail.

4.1 Sequence-to-sequence (seq2seq)

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

4.1.1 Recurrent encoder and decoder

As described in Bahdanau 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.

4.1.2 Model 0: Standard Attention

In Bahdanau 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*.

4.1.3 Model 1 and 2: Coarse-to-Fine Soft Attention

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*.

4.2 Model 3: Coarse-to-Fine Sparse 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., 2015), 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*.

4.2.1 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.

4.2.2 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.

4.3 Sparsemax

Chapter 5

Experiments

5.1 Data

5.1.1 Yuntian's stuff

5.1.2 CNN/Dailymail

Experiments were performed on the CNN/Dailymail dataset from **?**. 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 5.1 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 5.1: Statistics for CNN/Dailymail data.

5.2 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.

5.3 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.

5.4 Models

5.4.1 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.

5.4.2 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 $k = 5$.

5.5 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 6

Results

6.1 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 6.1: Summarization results for CNN/Dailymail.

6.2 Analysis

See Table 6.1 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 7

Discussion

Chapter 8

Conclusion

Bibliography

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

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.

Yoshua Bengio, Nicholas Léonard, and Aaron C Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3, 2013. URL http://arxiv.org/abs/1308.3432.

Pf Brown, Vjd Pietra, S Pietra, and R Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993. ISSN 08912017. doi: 10.1080/08839514.2011.559906. URL http://www.aclweb.org/anthology/J93-2003.

J Carbonell and J Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 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.

Danqi Chen, Jason Bolton, and Christopher D Manning. A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. *Acl 2016*, pages 2358–2367, 2016.

James Clarke and Mirella Lapata. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429, 2008.

Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 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. *Journal of Machine Learning Research*, 12:2493–2537, 2011. ISSN 0891-2017. doi: 10.1.1.231.4614.

Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, Yoshua Bengio, Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes. *CoRR*, abs/1607.0:13, 2016. doi: 10.1051/0004-6361/201527329. URL http://arxiv.org/abs/1607.00036.

KM Hermann, T Kocisky, and E Grefenstette. Teaching machines to read and comprehend. *Advances in 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 computation*, 9(8): 1735–1780, 1997.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (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. *Artificial Intelligence*, 139(1):91–107, 2002.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Y LeCun and Y Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(April 2016):255–258, 1995. ISSN 1098-7576. doi: 10.1109/IJCNN.2004.1381049. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.9297{\&}rep=rep1{\&}type=pdf.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A Hierarchical Neural Autoencoder for Paragraphs and Documents. *CoRR*, abs/1506.0, 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 preprint arXiv:1603.06155*, 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 summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 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. *Proceedings of The 33rd International Conference on Machine Learning*, pages 1614–1623, 2016. URL http://arxiv.org/abs/1602.02068.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. *Proceedings of NAACL-HLT*, pages 1–11, 2016.

T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 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. *Advances in Neural Information Processing Systems*, pages 2204—-2212, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 0028-0836. doi: 10.1038/nature14236. URL http://dx.doi.org/10.1038/nature14236.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Proceedings of CoNLL*, abs/1602.0:280–290, 2016. URL http://arxiv.org/abs/1602.06023.

Ani Nenkova and Kathleen McKeown. Automatic Summarization. *Foundations and Trends® in Information Retrieval*, 5(3):235–422, 2011. ISSN 1554-0669. doi: 10.1561/1500000015. URL http://www.nowpublishers.com/product.aspx?product=INR{\&}doi=1500000015.

Paul Over, Hoa Dang, and Donna Harman. DUC in context. *Information Processing & Management*, 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, *Advances in Neural Information Processing Systems 29*, pages 3621–3629. Curran Associates, Inc., 2016.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. Unsupervised Pretraining for Sequence to Sequence Learning. *CoRR*, 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. *CoRR*, abs/1511.0:1–15, 2015. doi: 10.1371/journal.pcbi.1005055. URL http://arxiv.org/abs/1511.06732.

F. Rosenblatt. A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471(Electronic);0033-295X(Print). doi: 10.1037/h0042519.

Alexander M Rush, Sumit Chopra, and Jason Weston. A Neural Attention Model for Abstractive Sentence Summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 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*, pages 1–13, 2015. ISSN 10495258. URL http://arxiv.org/abs/1506.05254.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: the Sparsely-Gated Mixture-of-Experts Layer. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Dou Shen, Jian-tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document Summarization using Conditional Random Fields. *Science*, 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.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961. URL http://dx.doi.org/10.1038/nature16961.

S Sukhbaatar, J Weston, and R Fergus. End-to-end memory networks. *Nips*, pages 1–9, 2015. URL http://papers.nips.cc/paper/5846-end-to-end-memory-networks.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Krysta Marie Svore, Lucy Vanderwende, Christopher J C Burges, K Svore Vanderwende, L., and Burges, C., and K Svore Vanderwende, L., and Burges, C. Enhancing single-document summarization by combining RankNet and third-party sources. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 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. *Machine learning*, 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, Ruslan Salakhutdinov Richard S. Zemel Yoshua Bengio, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML*, 14:77—–81, 2015. 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 *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119, 2004.

Wojciech Zaremba and Ilya Sutskever. Reinforcement Learning Neural Turing Machines. *CoRR*, abs/1505.0, 2015. URL http://arxiv.org/abs/1505.00521.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in 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.

.1 Attention Visualizations

hi