Abstractive Text Summarization

Using Seq2Seq Attention Models

Soumye Singhal Prof. Arnab Bhattacharya

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

22^{nd} November, 2017

Outline

The Problem

- Why Text Summarization
- Extractive vs Abstractive

Baseline Model

- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements

- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Outline

The Problem

- Why Text Summarization
- Extractive vs Abstractive

Baseline Model

- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements

- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Why Text Summarization?

- In the modern Internet age, textual data is ever increasing
- Need some way to condense this data while preserving the information and meaning.
- Text summarization is a fundamental problem that we need to solve.
- Would help in easy and fast retrieval of information.

Outline

- The Problem
  - Why Text Summarization
  - Extractive vs Abstractive
- Baseline Model
  - Vanilla Encoder-Decoder
  - Attention is all you need!
- Metrics and Datasets
- Improvements
  - Hierarchical Attention
  - Pointer-Generator Network
  - Coverage Mechanism
  - Intra-Attention
  - Reinforcement Based Training
- Challenges and Way Forward

Extractive vs Abstractive

- **Extractive summarization**
  - Copying parts/sentences of the source text and then combine those part/sentences together to render a summary.
  - Importance of sentence is based on linguistic and statistical features
- **Abstractive summarization**
  - These methods try to first understand the text and then rephrase it in a shorter manner, using possibly different words
  - For perfect abstractive summary, the model has to first truly understand the document and then try to express that understanding in short possibly using new words and phrases.
  - Much harder than extractive.
  - Has complex capabilities like generalization, paraphrasing and incorporating real-world knowledge.

Deep Learning

- Majority of the work has traditionally focused on Extractive approaches due to the easy of defining hard-coded rules to select important sentences than generate new ones.
- But they often don't summarize long and complex texts well as they are very restrictive.
- The traditional rule-based AI does poorly on Abstractive Text Summarization.
- Inspired by the performance of Neural Attention Model in the closely related task of Machine Translation Rush et al. 2015 and Chopra et al. 2016 applied this Neural Attention Model to Abstractive Text Summarization and found that it already performed very well and beat the previous non-Deep Learning-based approaches.

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Recurrent Neural Network

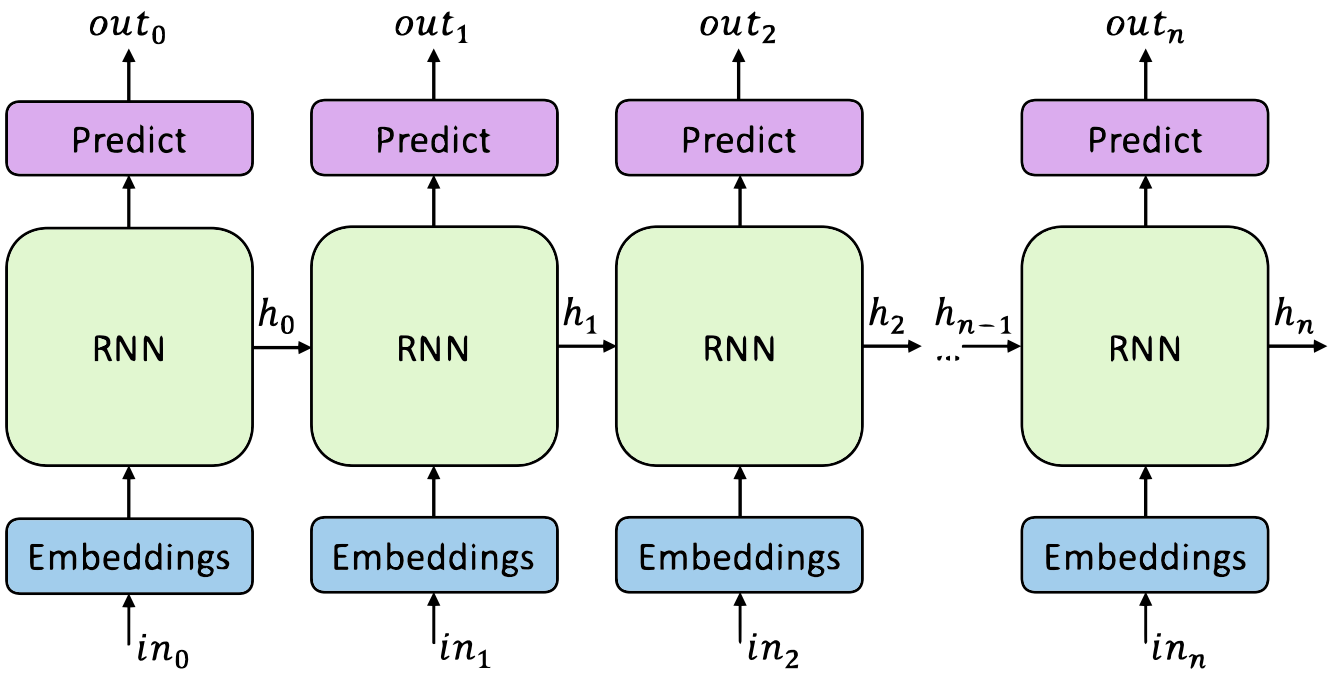
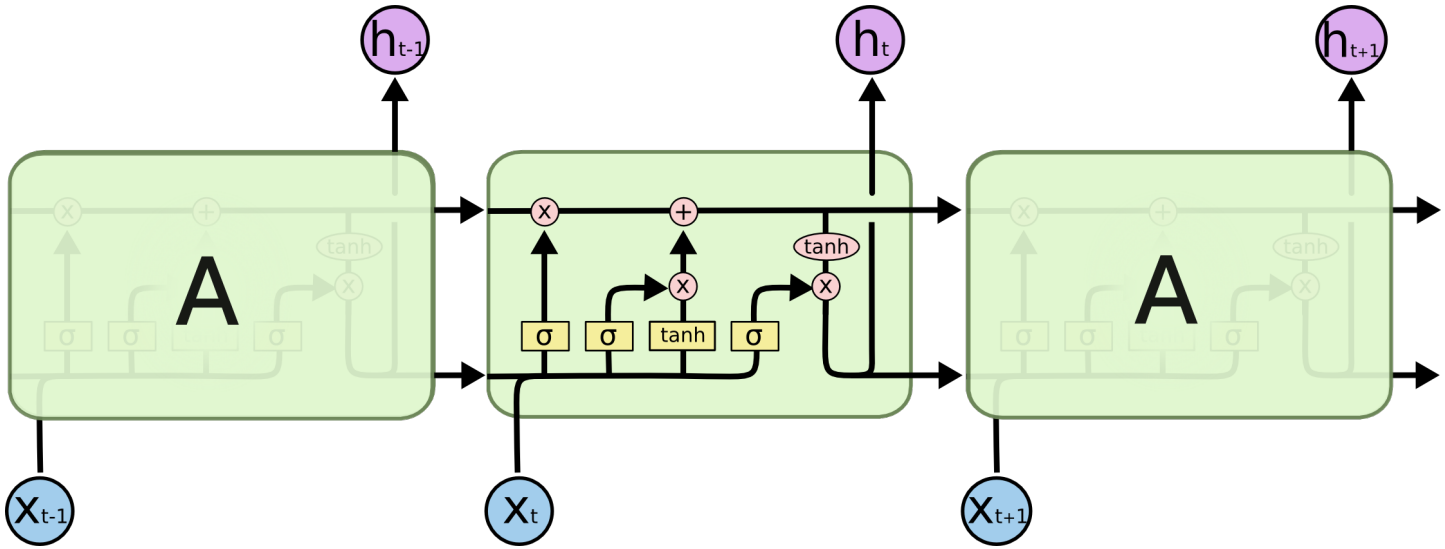


Figure: An unrolled RNN

- w_i - input tokens of source article
- h_i - Encoder hidden states
- $P_{vocab} = softmax(Vh_i + b)$ is the distribution over vocabulary from which we sample out_i

Long-Short Term Memory



- If the context of the word is far away, RNN's struggle to learn.
- Vanishing Gradient Problem
- LSTMs selectively pass and forget information.

[1] Image taken from `colah.github.io`

Long-Short Term Memory

Forget Gate Layer

- $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
- $C_t = C_t \otimes f_t$

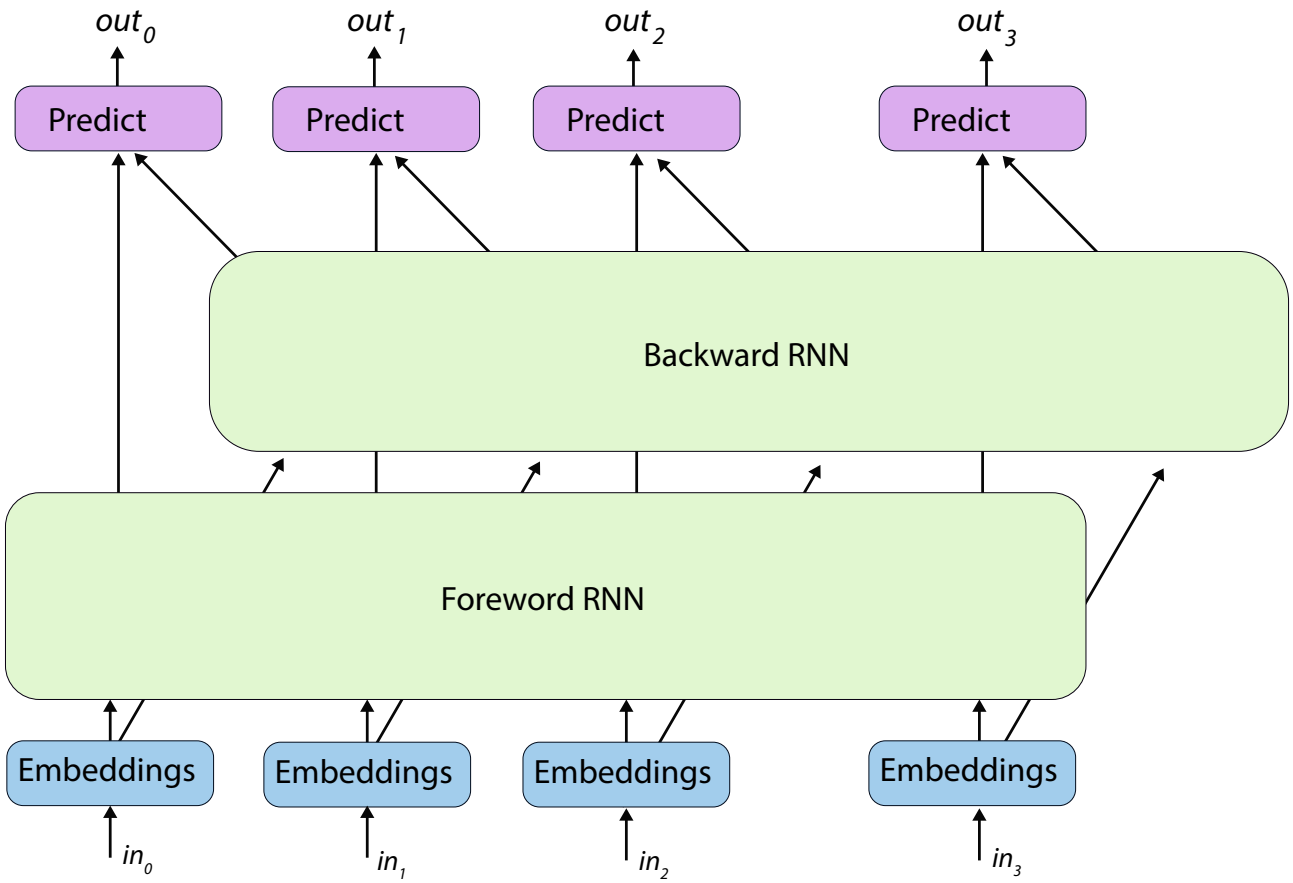
Input Gate Layer

- $i_t = \sigma(W_t[h_{t-1}, x_t] + b_i)$
- $\widetilde{C_t} = tanh(W_i[h_{t-1}, x_t] + b_c)$
- $C_t = \widetilde{C_t} \otimes t_t + C_t$

Output Gate Layer

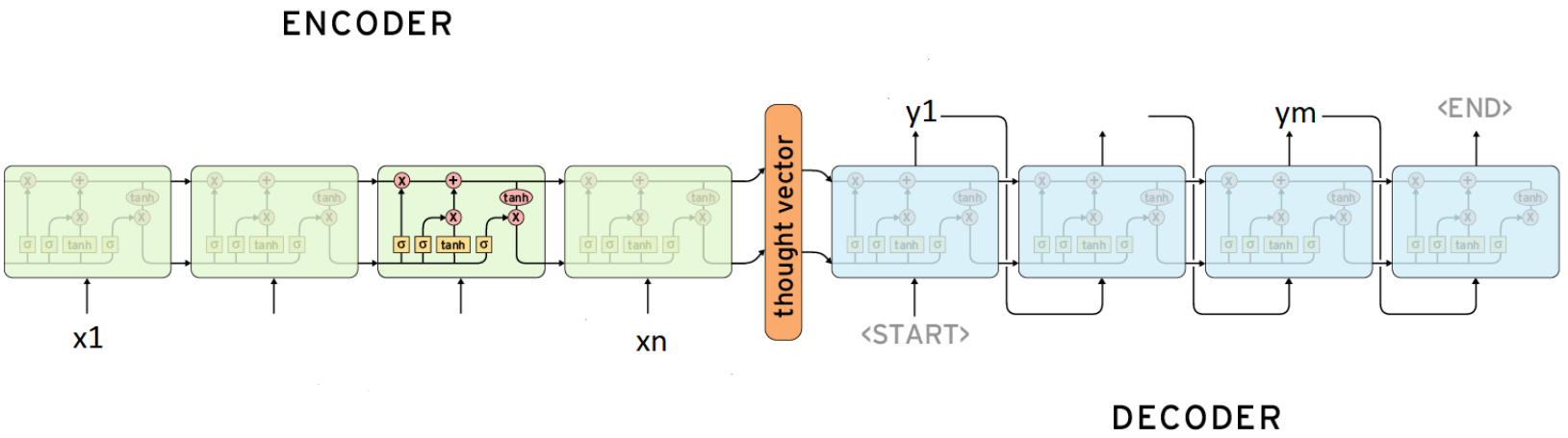
- $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
- $h_t = o_t * tanh(C_t)$

Bi-Directional RNN



- Two passes on source computing hidden states $\overleftarrow{h_t}$ and $\overrightarrow{h_t}$
- $h_t = [\overleftarrow{h_t}, \overrightarrow{h_t}]$ now encodes past and future information.

Vanilla Encoder-Decoder



[2]

- It consists of an Encoder(Bidirectional LSTM) and a Decoder LSTM network.
- The final hidden state from the Encoder(thought vector) is passed into the Decoder.

[2] Image taken from `colah.github.io`

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

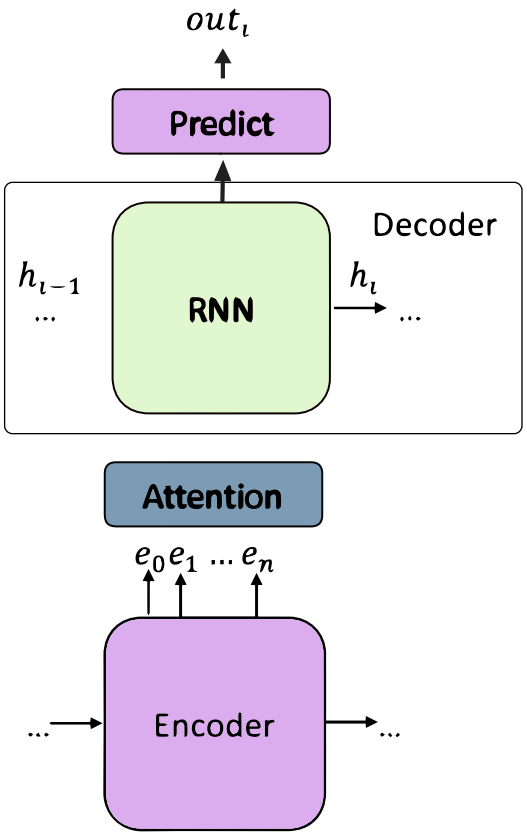
Challenges and Way Forward

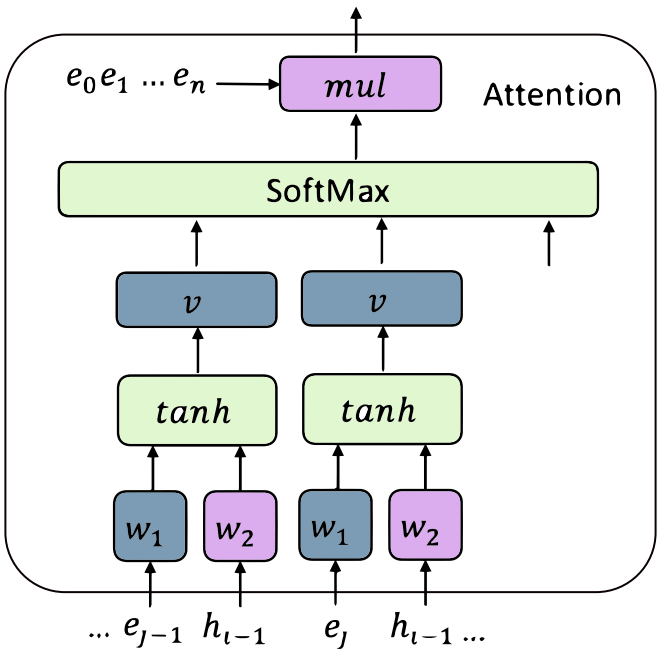
Why do we need Attention?

- The basic encoder-decoder model fails to scale up.
- The main bottleneck is the fixed sized thought vector
- Not able to capture all the relevant information of the input sequence as the model sizes up.
- At each generation step, only a part of the input is relevant.
- This is where attention comes it.
- It helps the model decide which part of the input encoding to focus on at each generation step to generate novel words.
- At each step, the decoder outputs hidden state h_i, from which we generate the output.

Attention is all you need!

- $importance_{it} = V * tanh(e_i W_1 + h_t W_2 + b_{attn})$.
- Attention Distribution $a^t = softmax(importance_{it})$
- ContextVector $h_t^* = \sum_i e_i * a_i^t$





[3] Image stylized from https://talbaumel.github.io/attention/

Training

- Context Vector is then fed into two layers to generate distribution over the vocabulary from which we sample.
- $P_{vocab}(w) = softmax(V'(V[h_t, h_t^*] + b) + b')$
- For the loss at time step t, $loss_t = -\log P(w_t*)$, where w_t^* is the target summary word.
- $LOSS = \frac{\sum_{t=0}^{T} loss_t}{T}$
- We then use the Backpropagation Algorithm to get the gradient and learn the parameters

Generating the Summaries

At each step, the decoder outputs a probability distribution over the target vocabulary. To get the output word at this step we can do the following

- Greedy Sampling, ie choose the mode of the Distribution
- Sample from the distribution.
- **Beam Search** - Choosing the top k most likely target words and then feeding them all into the next decoder input. So at each time-step t the decoder gets k different possible inputs. It then computes the top k most likely target words for each of these different inputs. Among these, it keeps only the top-k out of k^2 and rejects the rest. This process continues. This ensures that each target word gets a fair shot at generating the summary.

Metrics

- If target summary is not given
  - Need a similarity measure between summary and source document.
  - In a good summary, the topics covered would be similar
  - Use topic models like Latent Semantic Analysis(LSA) and Latent Dirichlet Allocation(LDA)
- If the target summary is given
  - Use metrics like `ROUGE`(Lin 2004) and `METEOR`
  - They are essentially string matching metrics
  - `ROUGE-N` measures the overlap of N-grams between the system and reference summary
  - `ROUGE-L` is based on longest common subsequences. Takes into account sentence level similarity.
  - `ROUGE-S` is the skip-gram variant

Dataset

Sentence level Datasets

- DUC-2004
- Gigaword

Large-Scale Dataset by Nallapati et al. 2016

- CNN/Daily Mail Dataset adapted for summarization.

Problems with Baseline

Though the baseline gives decent results, they are clearly plagued by many problems

- They sometimes tend to reproduce factually incorrect details.
- Struggles with Out of Vocabulary (OOV) words.
- They are also a bit repetitive and focus on a word/phrase multiple times.
- Focus mainly on single sentence summary tasks like headline generation.

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Feature-rich Encoder

- Introduced by Nallapati et al. 2016
- Aim is to input more more information about the source text into encoder
- Apart from word-embeddings like `word2vec`, `GloVe` also incorporate more linguistic features like
  - POS(parts of speech) tags
  - named-entity tags
  - TF-IDF statistics
- Though it speeds up training, it hurts the abstractive capabilities of the model.

Hierarchical Attention

- Introduced by Nallapati et al. 2016.
- For bigger source document, they try to also identify key **sentences** for the summary.
- Two Bi-Direction RNN at source text
  - One at word level
  - Another at sentence level
  - Word level attention is then weighted by corresponding sentence level attention.

$$P^a(j) = \frac{P^a_w(j)P^a_s(s(j))}{\sum_{k=1}^{N_d} P^a_w(k)P^a_s(s(k))}$$

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

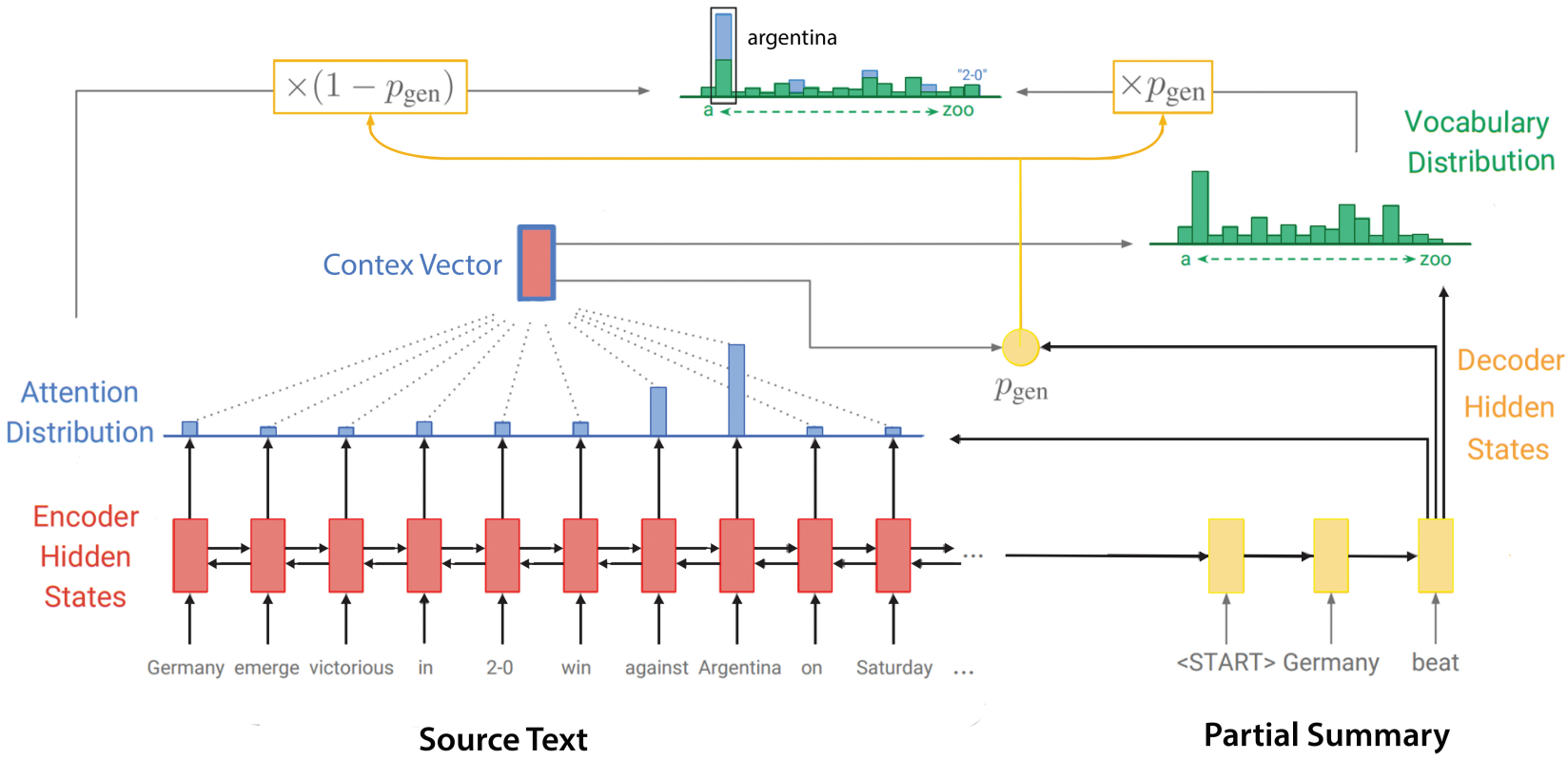
Challenges and Way Forward

Pointer-Generator Network

Introduced by See et al. 2017.

- Helps to solve the challenge of OOV words and factual errors.
- Works better for multi-sentence summaries.
- Ides is to choose between generating a word from the fixed vocabulary or copying one from the source document at each step of the generation.
- It brings in the power of extractive methods by pointing (Vinyals et al. 2015)
- So for OOV words, simple generation would result in UNK, but here the network will copy the OOV from the source text.

Pointer-Generator Network



[4] Image taken from blog, www.abigailsee.com

Pointer-Generator Network

- At each step we calculate generation probability p_{gen}
- $p_{gen} = \sigma(w_{h*}^T h_t^* + w_s^T h_t + w_x^T x_t + b_{ptr})$
- x_t is the decoder input.
- Parameter $w_{h^*}, w_s, w_x, b_{ptr}$ are learnable.
- Now this p_{gen} is used as a switch.
- $P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t$
- Note that for OOV word $P_{vocab}(w) = 0$, so we end up pointing.

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Coverage Mechanism

- The cause of repetitiveness of the model can be accounted for by increased and continuous attention to a particular word.
- So we can use `Coverage Model` by Tu et al. 2016.
- Coverage Vector $c^t = \sum_{t'=0}^{t-1} a^{t'}$
- Intuitively, by summing the attention at all steps we are keeping track of how much coverage each encoding, e^i has received.
- Now, give this as input to attention mechanism.
- $importance_{it} = V * tanh(e_i W_1 + h_t W_2 + W_c c_i^t + b_{attn})$
- Penalize attending to things that have already been covered.
- $covloss_t = \sum_i min(a_i^t, c_i^t)$ penalizes overlap between attention at this step and coverage till now.
- $loss_t = -\log P(w_t*) + \lambda covloss_t$

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

Intra-Attention

- Traditional approaches attend on the encoder states.
- But the current word being generated also depends upon what previous words were generated.
- So Paulus et al. 2017 used Intra-Attention on Decoder outputs.
- This approach also avoids repeating things.
- Decoder context vector c_t^* is generated in a similar way to encoder attention.
- c_t^* passed on to generate $P_{vocab}(w)$

Outline

The Problem
- Why Text Summarization
- Extractive vs Abstractive

Baseline Model
- Vanilla Encoder-Decoder
- Attention is all you need!

Metrics and Datasets

Improvements
- Hierarchical Attention
- Pointer-Generator Network
- Coverage Mechanism
- Intra-Attention
- Reinforcement Based Training

Challenges and Way Forward

How to correct my mistakes?

- During training, we always feed in the correct inputs to the decoder, no matter what the output was at the previous step.
- Model doesn't learn to recover from its mistakes.
- It assumes that it will be given the golden token at each step in the decoding.
- During testing if the model produces even one wrong word then the recovery is hard.

A naive way to do rectify this problem is that during training, toss a coin with $\mathbb{P}[heads] = p$ to decide between choosing generated output from the previous step or taking the golden token.

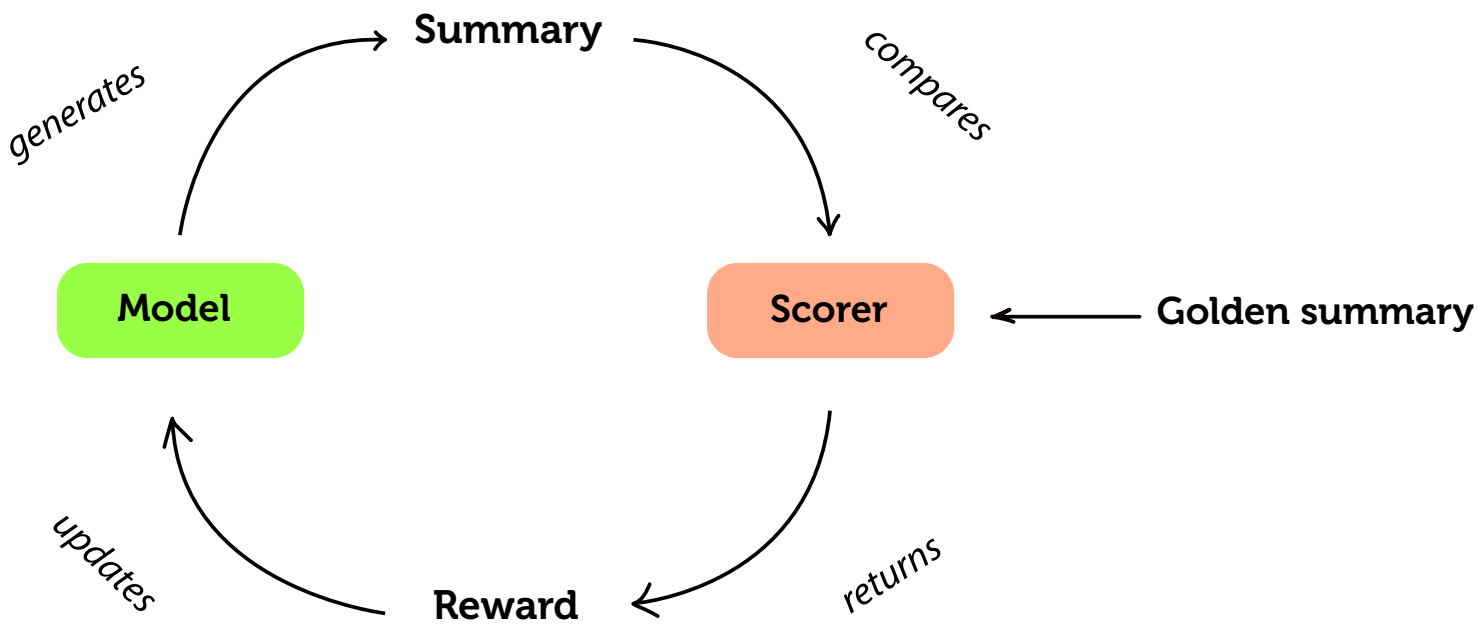
Training using Reinforcement Learning

- There are various ways in which the document can be effectively summarized. The reference summary is just one of those possible ways.
- There should be some scope for variations in the summary

This is the idea behind Reinforcement based learning introduced by Paulus et al. 2017 which gave significant improvement over the baseline. This is the current state of the art.

- During training, we first let the model generate a summary using its own decoder outputs as inputs.
- After the model produces its own summary, we evaluate the summary in comparison to the reference summary using the `ROUGE` metric.
- We then define a loss based on this score. If the score is high that means the summary is good and hence the loss should be less and vice-versa.

Training using Reinforcement Learning



Policy Learning

- We use self-critical Policy gradient training.
- We generate two strings y^s and $\hat{y}$
- $y^s \sim \mathbb{P}(y_t^s | y_1^s, \cdots, y_{t-1}^s, x)$ ie sampling and $\hat{y}$ by greedy search.
- y^* is the ground truth.
- $r(y)$ is the reward for sequence y compared with $y*$.
- $L_{rl} = (r(\hat{y}) - r(y^s)) \sum_t \log \mathbb{P}(y_t^s | y_1^s, \cdots, y_{t-1}^s, x)$.

Problems in Training using Reinforcement Learning

- It's possible to achieve a very high ROUGE score, without the summary being human readable.
- Reflects that ROUGE doesn't exactly capture the way we humans evaluate summary.
- Now, since the above method optimizes for the ROUGE scores, it may produce summaries with very high ROGUE scores, but which are barely human-readable.
- So to curb this problem, we train our model in a mixed fashion using both Reinforcement learning and Supervised Training.
- We can interpret it as, RL training giving the summary a global sentence/summary level supervision and Supervised training giving a local word level supervision.
- $L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{ml}$

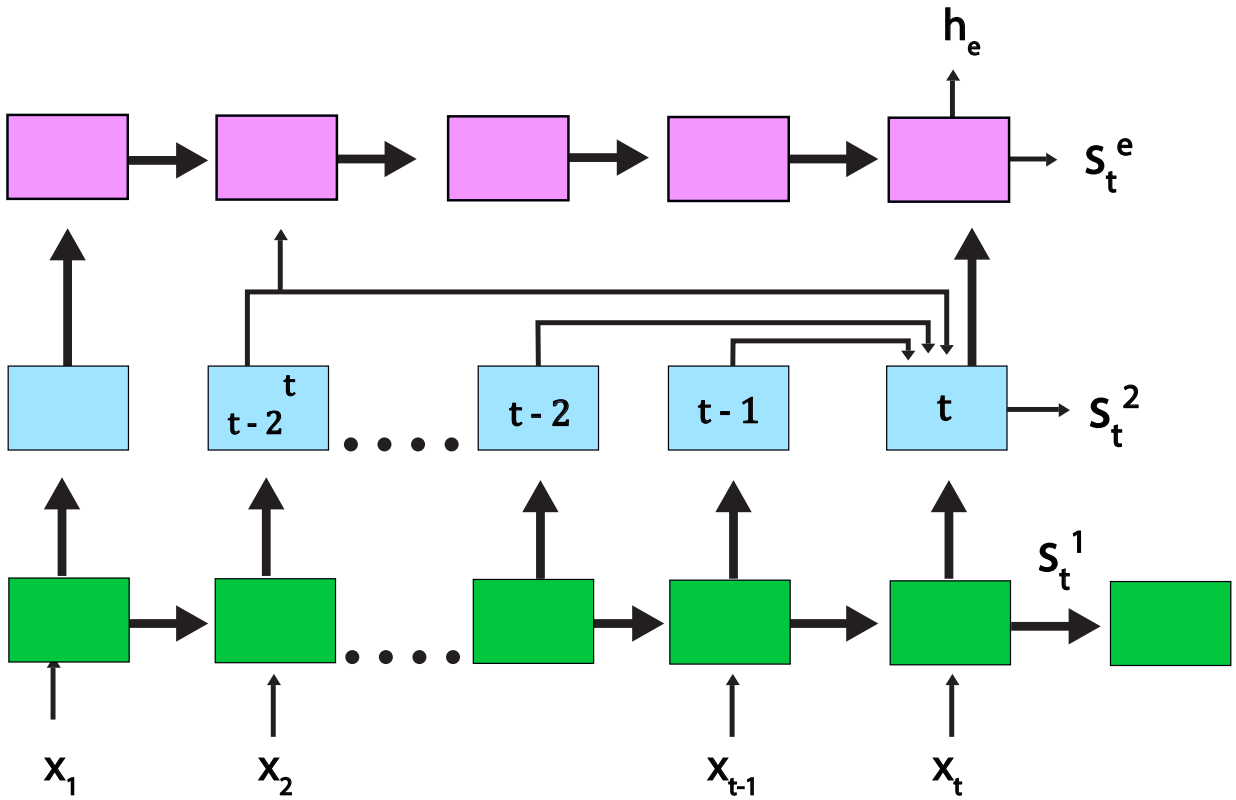
Challenges

- As pointed out by Paulus et al. 2017, `ROUGE` as a metric is deficient.
- Dataset issues
  - A majority of the dataset that is available is news dataset.
  - Can come up with a good summary only by looking at the top few sentences.
  - All the above-discussed models discussed above assume this and look at only the top 5-6 sentences of the source article.
  - Need a richer dataset for multi-sentence Text Summarization.
- Scalability Issues - the Multi-sentence problem largely unsolved.
- Need a lot of data and computation power.

Future Work

- To solve the problem of ROUGUE metric in the Reinforcement Learning based training method, we can instead learn a Discriminator separately first, which given a document and corresponding summary tells how good the summary it.
- The problem of long document summarization has two main problems
  - Vanishing Gradient Problem
  - LSTM's help information pass along further
  - But, the errors don't propagate further back in time well.
  - Maximum 20-25 steps only.
  - **Logarithmic Residual LSTM's**

Logarithmic Residual LSTMs



References I


Chopra, Sumit et al. (2016). "Abstractive sentence summarization with attentive recurrent neural networks". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98.


Lin, Chin-Yew (2004). "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out: Proceedings of the ACL-04 workshop*. Vol. 8. Barcelona, Spain.


Nallapati, Ramesh et al. (2016). "Abstractive text summarization using sequence-to-sequence rnns and beyond". In: *arXiv preprint arXiv:1602.06023*.


Paulus, Romain et al. (2017). "A Deep Reinforced Model for Abstractive Summarization". In: *arXiv preprint arXiv:1705.04304*.

References II

- Rush, Alexander M et al. (2015). "A neural attention model for abstractive sentence summarization". In: *arXiv preprint arXiv:1509.00685*.
- See, Abigail et al. (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *arXiv preprint arXiv:1704.04368*.
- Tu, Zhaopeng et al. (2016). "Modeling coverage for neural machine translation". In: *arXiv preprint arXiv:1601.04811*.
- Vinyals, Oriol et al. (2015). "Pointer networks". In: *Advances in Neural Information Processing Systems*, pp. 2692–2700.