Automatic Text Summarization

Abeer Ahmad (38)
Mohammed Deifallah (59)

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

Problem Definition

Text summarization is the process of creating a short and coherent version of a longer document. For example:

- **Input:** the sri lankan government on wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country.
- **Output:** sri lanka closes schools as war escalates

Followed Technique

In our problem, we followed the single-document generic abstractive approach:

- Single-Document → input length is short.
- Generic → model makes no assumption about the domain of input text - most common so far.
- Abstractive → model forms its own phrases and sentences to offer a more coherent summary - more appealing, but much more difficult than extractive summarization.

Dataset

- CNN, part of DeepMind Q&A dataset was our first choice. This dataset contains different CNN articles, supplemented with multiple highlights.
- We then sticked to a preprocessed version of the Gigaword dataset, a larger (4x) but easier-to-handle dataset than the CNN; as it has its ground truth labels as single-sentence summaries.
    - **Article:** “`south korea 's nuclear envoy kim sook urged north korea monday to restart work to disable its nuclear plants and stop its `` typical '' brinkmanship in negotiations #`”
    - **Ground Truth Summary:** “`envoy urges north korea to restart nuclear disablement`”
- This way, it is more like ***“Headline Generation”*** problem.

Evaluation Metric

- Nearly all approaches taken to tackle similar problems use ROUGE as an evaluation metric.
- We used the same metric for our model evaluation:
  - ROUGE-1
  - ROUGE-2
  - ROUGE-L

State-of-the-Art Results

	ROUGE-1	ROUGE-2	ROUGE-L
Pretraining-Based Model (2019)	**41.71**	**19.49**	**38.79**
GAN Model (2018)	39.92	17.65	36.71
Pointer-Generator Model (2017)	39.53	17.28	36.38

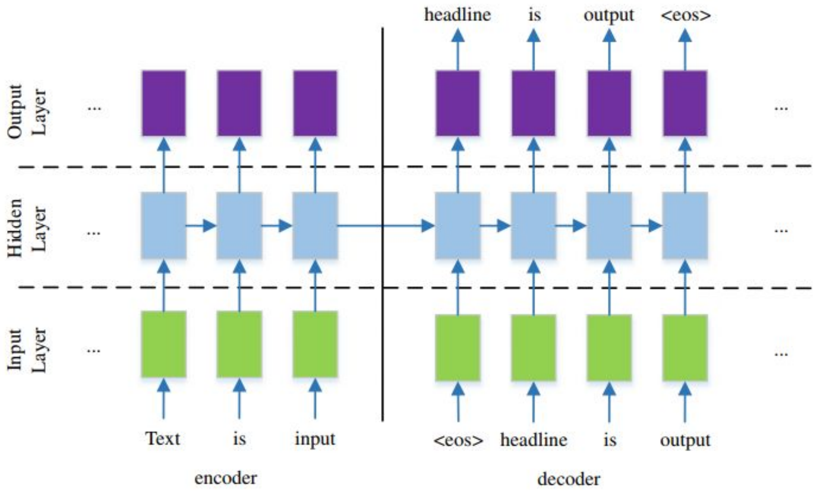
- These results are all based on text summarization models trained over the CNN dataset.

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

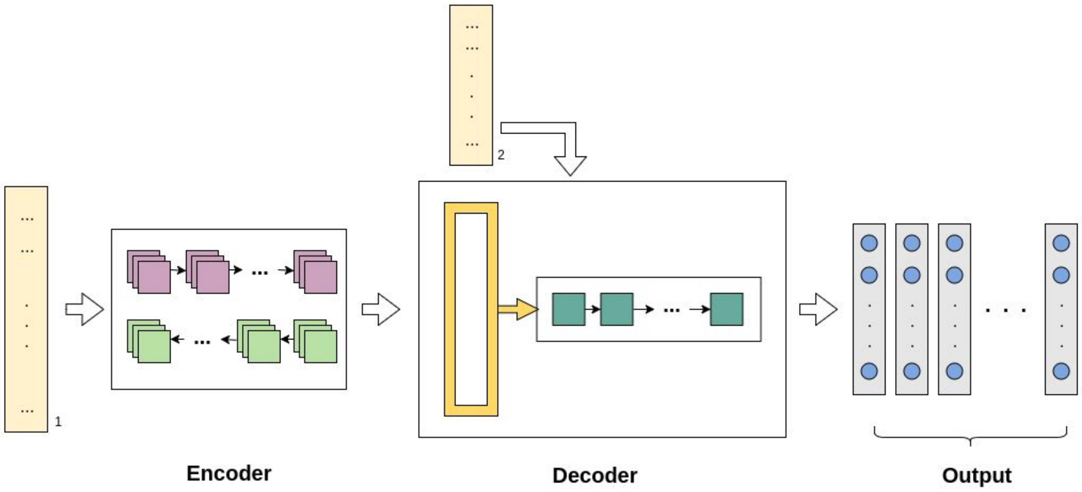
Model Architecture

Encoder-Decoder Seq2Seq Network



Source

Model Architecture - A Closer Look



- Input Embeddings for Both Articles [1] and Summaries [2]
- Stacked Forward LSTM
- Stacked Backward LSTM
- Attention Layer
- LSTM
- Dense Layer (Output)

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

Training

Hyperparameters

- number of stacked LSTM layers (default 2)
- number of LSTM hidden units (default 150)
- emeddings
  - learned VS. pretrained
  - size (default 300)
- batch size (default 64)
- learning rate (default 0.01)
- dropout keep-probability (default 0.8)

Loss Function

- sparse softmax cross entropy

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

Evaluation

- To evaluate our model, we first trained it under various combinations of hyperparameter values that we anticipated would make significant difference in the obtained results.
- For each trained model, we used it to generate summaries for our test set articles, and evaluated the resulting summaries compared to the ground truth titles using ROUGE.
- Following are some of the trials we performed...

Evaluation

- pretrained GloVe embeddings
- 512-sample batch
- 3 epochs
- others -- default

- Average Training Time per Epoch:
  - 3 hours
- Results:
  - ROUGE-1: 31
  - ROUGE-2: 12.53
  - ROUGE-L: 27.27

Evaluation

- 200-feature self-learned embeddings
- 100-unit LSTM
- 64-sample batch
- 1 epoch
- others -- default

- Average Training Time per Epoch:
  - 5 hours
- Results:
  - ROUGE-1: 19.77
  - ROUGE-2: 6.02
  - ROUGE-L: 16.7

Evaluation

- pretrained GloVe embeddings
- 4-layer stacked LSTM
- 512-sample batch
- 3 epochs
- others -- default

- Average Training Time per Epoch:
  - 3 hours

- Results:
  - ROUGE-1: 19.37
  - ROUGE-2: 5.55
  - ROUGE-L: 17.02

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

Best Result

ROUGE

- ROUGE-1: 31
- ROUGE-2: 12.53
- ROUGE-L: 27.27

Model Configuration

- number of stacked LSTM layers (default 2)
- number of LSTM hidden units (default 150)
- emeddings
  - pretrained (GloVe)
  - size (default 300)
- batch size (512)
- learning rate (default 0.01)
- dropout keep-probability (default 0.8)

Outline

- Recap
  - Problem Definition
  - Followed Technique
  - Dataset
  - Evaluation Metric
  - State-of-the-Art Results
- Model Architecture
- Training
  - Hyperparameters
  - Loss Function
- Evaluation
- Best Result
- How to Extend?

How to Extend?

- Apply *“Beam Search”* technique for better sequence generation.
- Support *“Pointer-Generator”* method to tackle factual errors, OOV (out-of-vocab) words and repeating.
- Generalize decoder network to work on CNN and other datasets with multi-sentence summaries.

Thank You