**Abstractive Text Summarization**

Overview

First, we need to preprocess the data by constructing an embedding of the text. Embedding the input converts the text into numbers, a more interpretable numerical representation of the data for the encoder-decoder network to work with. I experimented with two different embedding methods: Word2Vec and Global-Vectors (GloVe). Word2Vec is algorithm that combines continuous bag of words and the Skip-gram model to generate word vector representations. GloVe is an unsupervised learning algorithm for obtaining vector representations for words, training from a dictionary of common words.

The encoder-decoder model is composed of multiple recurrent neural networks, one of which works as an encoder, and one as a decoder. The encoder converts an input document into a latent representation (a vector), and the decoder reads the latent input, generating a summary as it decodes. With encoder decoder structures, issues to consider include determining how to set the focus on the import sentences and keywords, how to handle novel or rare words in the document, how to handle incredibly long documents, and how to make summaries readable and flexible with a large vocabulary.

The encoder-decoder recurrent neural network architecture has been shown to be effective when applied to text summarization. The architecture involves two components: an encoder and a decoder. The encoder reads the entire input sequence and encodes it into an internal representation, often a fixed-length vector. The decoder reads the encoded input sequence from the decoder and generates the output sequence, which is the summary. Both the encoder and decoder sub-models are trained jointly, meaning their output feed into the other as input.

A close up of a logo

Description automatically generated

The encoder is a bidirectional LSTM recurrent neural network (RNN). RNNs can use their internal state (memory) to process sequences of inputs. LSTMs are capable of learning long term dependencies by storing long-term states and inputs in gated cell memory. The tokenized words of the text are fed one-by-one into the encoder, a single-layer bidirectional LSTM, producing a sequence of hidden states, which is a latent representation of the input. The decoder is a singlelayer unidirectional LSTM, which receives the word embedding of the previous word, and the embedding is transformed into a word representation, which is part of the summary. I used the one-shot encoder-decoder model, where the entire output sequence is generated in a one-shot manner, meaning the decoder uses the latent context vector alone to generate the output summary.

Using simple TensorFlow implementation of text summarization using seq2seq library.

I have implemented to train our own encode-decoder model by using the GIGAWORLD dataset from scratch by trying to imitate the first state-of-the art encoder-decoder model.

A close up of a map

Description automatically generated

Reference paper: <https://www.aclweb.org/anthology/N16-1012/>

**Model**

Encoder-Decoder model with attention mechanism.

**Word Embedding**

Used [Glove pre-trained vectors](https://nlp.stanford.edu/projects/glove/) to initialize word embedding.

**Encoder**

Used LSTM cell with stack\_bidirectional\_dynamic\_rnn.

**Decoder**

Used LSTM BasicDecoder for training, and BeamSearchDecoder for inference.

**Attention Mechanism**

Used BahdanauAttention with weight normalization.

**Requirements**

* Python 3
* Tensorflow (>=1.8.0)
* pip install -r requirements.txt

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | gensim==3.3.0 | | wget==3.2 | | nltk==3.2.5 | |

**Usage**

Prepare data

$ python prep\_data.py

To use Glove pre-trained embedding, download it via

$ python prep\_data.py --glove

**Train**

We used sumdata/train/train.article.txt and sumdata/train/train.title.txt for training data. To train the model, use

$ python train.py

To use Glove pre-trained vectors as initial embedding, use

$ python train.py --glove

Additional Hyperparameters

|  |
| --- |
| $ python train.py -h  usage: train.py [-h] [--num\_hidden NUM\_HIDDEN] [--num\_layers NUM\_LAYERS]  [--beam\_width BEAM\_WIDTH] [--glove]  [--embedding\_size EMBEDDING\_SIZE]  [--learning\_rate LEARNING\_RATE] [--batch\_size BATCH\_SIZE]  [--num\_epochs NUM\_EPOCHS] [--keep\_prob KEEP\_PROB] [--toy]  optional arguments:  -h, --help show this help message and exit  --num\_hidden NUM\_HIDDEN  Network size.  --num\_layers NUM\_LAYERS  Network depth.  --beam\_width BEAM\_WIDTH  Beam width for beam search decoder.  --glove Use glove as initial word embedding.  --embedding\_size EMBEDDING\_SIZE  Word embedding size.  --learning\_rate LEARNING\_RATE  Learning rate.  --batch\_size BATCH\_SIZE  Batch size.  --num\_epochs NUM\_EPOCHS  Number of epochs.  --keep\_prob KEEP\_PROB  Dropout keep prob.  --toy Use only 5K samples of data |

**Test**

Generate summary of each article in sumdata/train/valid.article.filter.txt by

$ python test.py

It will generate result summary file result.txt. Check out ROUGE metrics between result.txt and sumdata/train/valid.title.filter.txt

#### **Sample Summary Output**

|  |
| --- |
| "general motors corp. said wednesday its us sales fell ##.# percent in december and four percent in #### with the biggest losses coming from passenger car sales ."  > Model output: gm us sales down # percent in december  > Actual title: gm december sales fall # percent  "japanese share prices rose #.## percent thursday to <unk> highest closing high for more than five years as fresh gains on wall street fanned upbeat investor sentiment , dealers said ."  > Model output: tokyo shares close # percent higher  > Actual title: tokyo shares close up # percent  "hong kong share prices opened #.## percent higher thursday on follow-through interest in properties after wednesday 's sharp gains on abating interest rate worries , dealers said ."  > Model output: hong kong shares open higher  > Actual title: hong kong shares open higher as rate worries ease  "the dollar regained some lost ground in asian trade thursday in what was seen as a largely technical rebound after weakness prompted by expectations of a shift in us interest rate policy , dealers said ."  > Model output: dollar stable in asian trade  > Actual title: dollar regains ground in asian trade  "the final results of iraq 's december general elections are due within the next four days , a member of the iraqi electoral commission said on thursday ."  > Model output: iraqi election results due in next four days  > Actual title: iraqi election final results out within four days  "microsoft chairman bill gates late wednesday unveiled his vision of the digital lifestyle , outlining the latest version of his windows operating system to be launched later this year ."  > Model output: bill gates unveils new technology vision  > Actual title: gates unveils microsoft 's vision of digital lifestyle |

**Limitations of this Abstractive summarization model**

Although abstractive summarization can be more intuitive and sounder like a human, I have found 3 limitations in the model:

* Firstly, training the model requires a lot of data and hence time. Although one can apply transfer learning here, pre-trained weights are not readily available and there is no open-source code that one can leverage to implement it
* An inherent problem with abstraction is that the summarizer reproduces factual details incorrectly. For instance, if the article talks about Germany beating Argentina 3–2, the summarizer may replace 3–2 by 2–0
* Repetition is another problem faced by the summarizer. As we can see in the second example above, some phrases are repeated in the summary
* Abstractive methods like the encoder-decoder network are capable of generating entirely new phrases and sentences to capture the meaning of the text. They tend to be more complex than extractive methods, since they learn to construct some cohesive phrasing of the relevant concepts. However, this also means they are more susceptible to error.

Metrics for result evaluation

BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially of a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summary (typically human-produced).

ROUGE-L – measures longest matching sequence of words using LCS. An advantage of using LCS is that it does not require consecutive matches, but in-sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, you don’t need a predefined n-gram length.

Reference:

<http://www.ccs.neu.edu/home/vip/teach/DMcourse/5_topicmodel_summ/notes_slides/What-is-ROUGE.pdf>

<http://text-analytics101.rxnlp.com/2017/01/how-rouge-works-for-evaluation-of.html>