ABSTRACTIVE SENTENCE SUMMARIZATION WITH SEQ-TO-SEQ MODELS

Raj Sundhar (rr966) **Abishek Prasanna** (ap1534)

5/7/18

533 Natural Language Processing
Prof. Matthew Stone

Abstract

Text summarization is a very challenging problem that can only be truly realized by understanding meaning of the textual content. Recently, deep recurrent neural networks have been used in the sequence to sequence framework to achieve good results on summarization tasks. In this project we explore the success achieved by a stacked LSTM encoder – attention based decoder architecture on the English Gigaword dataset. We employ the usual best practices of sequence to sequence models with a complete end-to-end training, and have decent results to show on generating summaries of 10 words or less, for 2 line texts with a maximum of 30 words. We also explore on improvements in network training time, computation and refinement of summaries through various experiments.

1. Introduction

Summarization is the task of obtaining a supplementary text for any given text that conveys the same meaning in fewer words. In Natural Language Processing, there are three main approaches to this problem: The first method preserves word order of the input text and deletes the least significant words to give summaries. The second and more prevalent method is that extractive summarization, which generates summaries with words in the input text alone but without any regard to word order. The third approach which is more similar to human-style summarization and obviously most challenging of the three, is that of abstractive summarization which produces summaries using any words without constraining to those from the input text.

Neural network models are able to achieve good abstractive summaries majorly because of its understanding of words and their similarity. The idea of distributional representation of words has been the primary facilitator for understanding word similarities. The distributional hypothesis claims that words that occur in the same contexts are similar, and has allowed to develop word vector models that encode the semantics and similarities of words that appear in a given text. Thus it can be seen that using an encoder-decoder model allows for abstractive summaries being generated at the decoder side because of the decoding component choosing one word for the other which are similar in some sense.

Sequence-to-Sequence models have been applied to attain state-of-art results on most NLP tasks which involve text generation. Using powerful LSTM and GRU based neural network layers has allowed to encode long term dependencies in input and target sequences. Until the inception of neural networks to NLP problems, Hidden Markov Models and chain models were applied to these problems, but they came with one major shortcoming, that of making several invalid independence assumptions about the dependencies of tokens in a sequence. LSTM and similar neural network models are able to encode and decode sequences without such assumptions and this is the primary reason for their qualitative efficiency in such problems.

O : Alice and Bob took the train to visit the zoo. They saw a baby giraffe, a lion, and a flock of colorful tropical birds.
E : *Alice and Bob visit the zoo. saw a flock of birds.*
A : Alice and Bob visited the zoo and saw animals and birds.

Figure 1 : Example of Extractive (E) and Abstractive (A) summaries for an input text O.

2. Background

There has been some significant work done on sentence summarization. The state-of-the-art models in the field in recent years have primarily been implemented by Alexander Rush and Sumit Chopra. In 2015, they demonstrated state-of-the art performance on sentence summarization task using feed forward window based neural network with an attention mechanism. In 2016, they used an attentive recurrent neural network to improve their results on the same task.

Despite their capability to capture long term dependency information, LSTM cells did not perform as expected with scaling in input size and thus attention mechanism was introduced to selectively store information and has since been adopted in most NLP sequence to sequence models with a significant improvement in result. Bahdanau's content based attention and Luong's global attention schemes are particularly simple and more popular. We adopt a dot attention mechanism from Luong's paper to implement our attentional decoder.

Decoding of words from indexes is achieved by greedy decoder and Beam search decoder. The greedy decoder outputs the most probable words at all times and generally does not give very good results. Beam Search Decoder takes a set beam size and at all times tracks the sequences with maximum posterior probability given by the product of individual probabilities of word occurrence. We go with a beam search decoder with beam size of 5 to get reasonable results. The working of Beam search decoder at generating character level word outputs is shown in Figure 3.

Today's state of the art approaches use more sophisticated models with Bidirectional LSTM cells wherein one layer of LSTM scans front to back and another LSTM layer scans back to front and both their outputs are concatenated to give a joint representation. These tend to be computationally intensive and in general increasing the number of units in a layer increases the computational requirements and training time exponentially.

Using more complex attention models like Pointer Generator networks is another avenue to improve decoding results. We start with the most simple dot attention model in this work and plan on extending in the future.

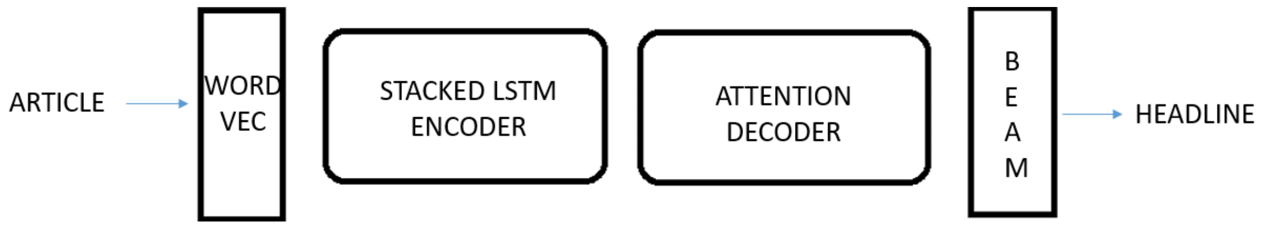


Figure 2 : Sequence to sequence Encoder-Decoder architecture. The preprocessed article is converted to GloVe word vector embedding model and are fed as input to the Stacked LSTM encoder which generates a thought vector. The attention decoder learns the attention weights to focus on to generate a summary fed as supervision for the network. The Beam Search decoder of beam size 10 allows for generation of sample summaries from index representation generated by the network.

The goal of the LSTM is to estimate the conditional probability $p(y_1,\dots,y_{T'}\,|x_1,\dots,x_T\,)$ where $(x_1,\dots,x_T\,)$ is an input sequence and $(y_1,\dots,y_{T'})$ is its corresponding output sequence whose length T' may differ from . The LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence $(x_1,\dots,x_T\,)$ given by the last hidden state of the LSTM, and then computing the probability of $(y_1,\dots,y_{T'})$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $(x_1,\dots,x_T\,)$:

$$p(y_1,\dots,y_{T'}\,|x_1,\dots,x_T\,) = \prod_{t=1}^{T'} p(y_t|v,y_1,\dots,y_{t-1})$$

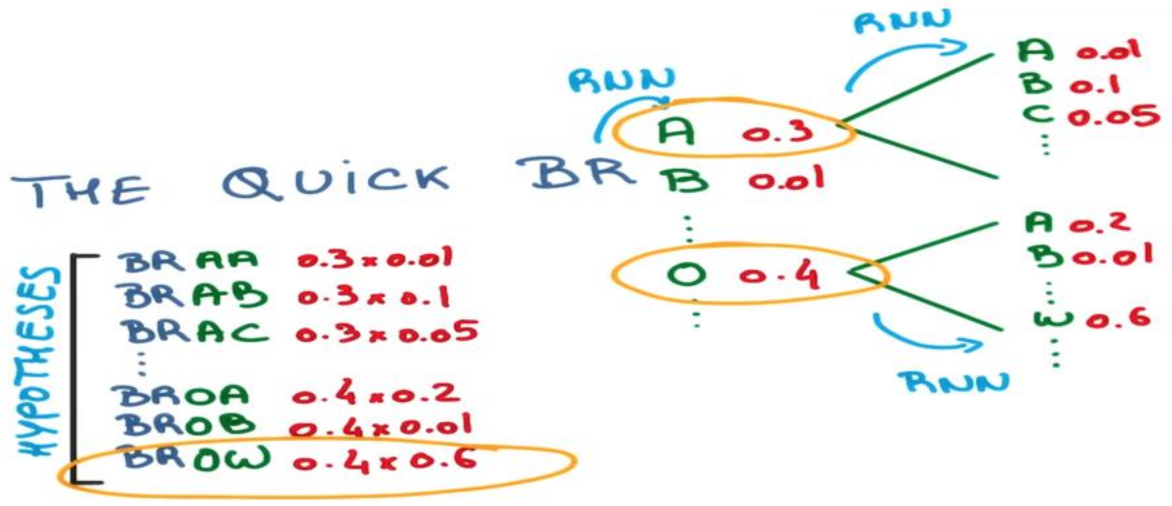


Figure 3 Beam Search at a character level. We follow a word level beam search to find the best fitting 10 sequences. The one with the maximum probability product that exceeds a certain threshold is given as the prediction result for novel data.

3. Methods

We filter out 50000 description-summary pairs from English Gigaword where with maximum of 25 and 10 words each. A vocabulary of 50000 most common words in this sample data set is taken and assigned Glove Word vector weights. In the remaining out of vocabulary words we search for those words for which we have Glove vectors and replace those specific out-of-vocab words with those words within the vocabulary with maximum (Glove) vector cosine similarity (above 0.5). Rest of the words are assigned $< unk >$ label. Input descriptions are prepadded with zeros and output summaries are postpadded. The input to network is concatenated sequence of description followed by summaries separated by $< eos >$ label. The output of the network is the summary alone postpadded after a $< eos >$ label to terminate the summary sequence. A model of our sequence to sequence summarizer is depicted in Figure 5.1.

An encoder model of 3 layers of 512 LSTM cells each is added on top of the embedding layer. A simple attention mechanism as given by Luong is tweaked slightly to produce our attention model. Instead of relying on the whole of the LSTM cells hidden representations for finding the similarity and estimating attention weights, we take only a small chunk of the topmost LSTM layers hidden state activations. This is given in Figure 5.2.

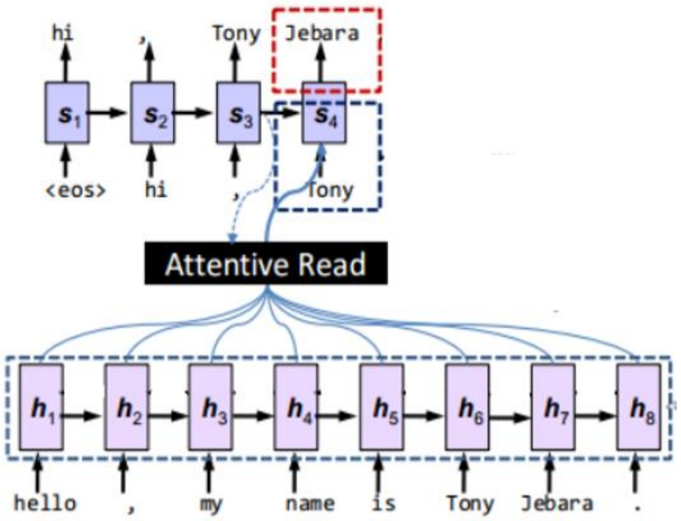


Figure 5.1: Sequence to Sequence summarizer model

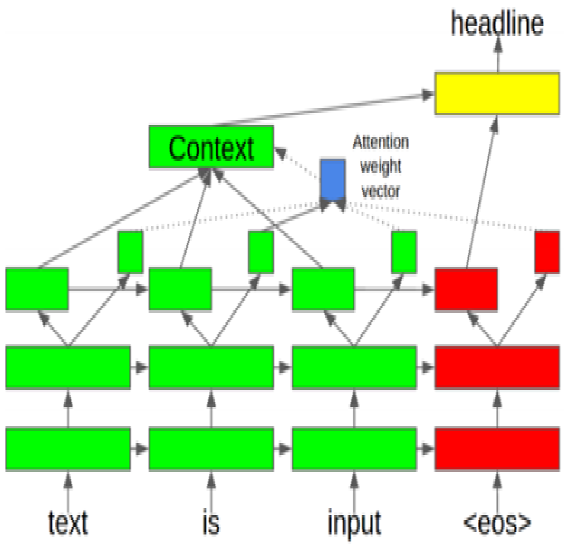
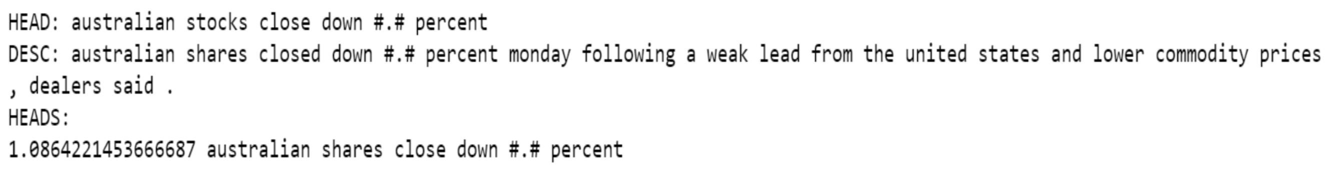


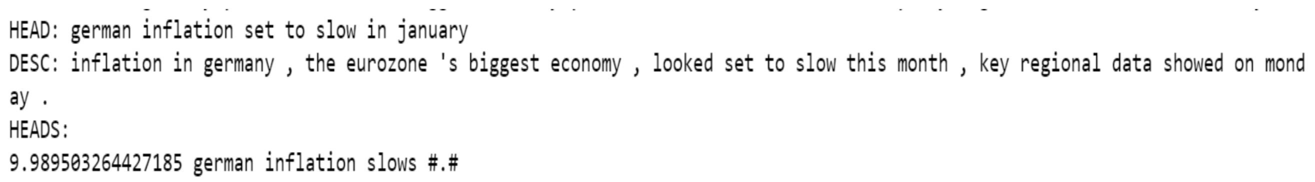
Figure 5.2 : Our attention model adapted from Luong’s Dot attention model

Unique Attention Model : The attention model aims to selectively focus on parts of the input sequence while generating summaries. We implement a global attention mechanism scheme that learns the context from all source words at all times. The other version called the local attention model attends only to a specific window size of the input. We use the first 40 LSTM cells (of the topmost layer) output activation to generate our Attention weight vector. The activation output from the remaining cells and the attention weight vector are dotted to obtain the Context vector which is later decoded by the beam search decoder after converting the vector to probabilities. At a high level what attention realizes is a learning of the word similarities between those of the input context and the required target word.



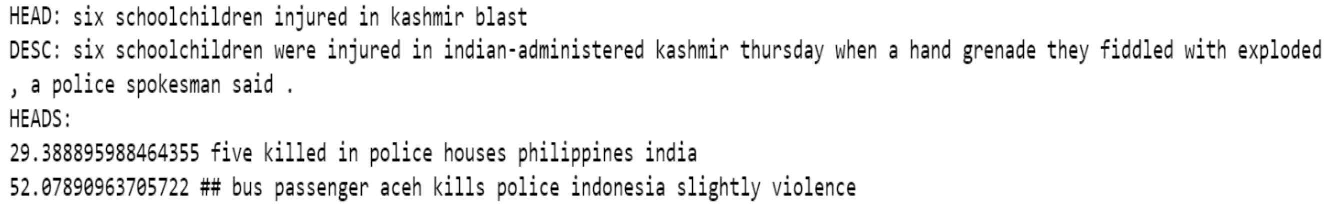
```
HEAD: australian stocks close down #.# percent
DESC: australian shares closed down #.# percent monday following a weak lead from the united states and lower commodity prices
, dealers said .
HEADS:
1.0864221453666687 australian shares close down #.# percent
```

Figure 4.1: Sample 1



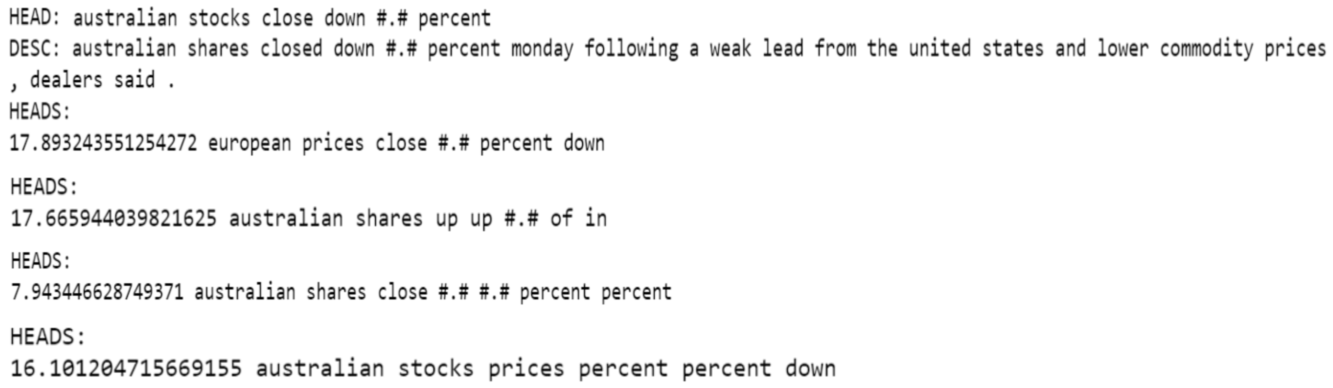
```
HEAD: german inflation set to slow in january
DESC: inflation in germany , the eurozone 's biggest economy , looked set to slow this month , key regional data showed on mond
ay .
HEADS:
9.989503264427185 german inflation slows #.#
```

Figure 4.2 : Sample 2



```
HEAD: six schoolchildren injured in kashmir blast
DESC: six schoolchildren were injured in indian-administered kashmir thursday when a hand grenade they fiddled with exploded
, a police spokesman said .
HEADS:
29.388895988464355 five killed in police houses philippines india
52.07890963705722 ## bus passenger aceh kills police indonesia slightly violence
```

Figure 4.3 : Sample 3



```
HEAD: australian stocks close down #.# percent
DESC: australian shares closed down #.# percent monday following a weak lead from the united states and lower commodity prices
, dealers said .
HEADS:
17.893243551254272 european prices close #.# percent down
HEADS:
17.665944039821625 australian shares up up #.# of in
HEADS:
7.943446628749371 australian shares close #.# #.# percent percent
HEADS:
16.101204715669155 australian stocks prices percent percent down
```

Figure 4.4 : Sample 4

Figure 4: Generated summaries for a few sample inputs from the test set. Samples 1 and 2 show very accurate summaries being generated. For stocks and inflation related inputs our model almost always generated very accurate summaries, and this was due to the prevalence of samples talking about the same in the training set. Sample 3 is an example that shows how the generated abstractive summary might capture false information. Numbers (three, four etc) or name of cities and countries generally occur together in the embedding dimension captured by GloVe Vectors. This makes the decoder to spit out similar words as a substitution and ultimately leading to capturing false information. Sample 4 captures the progress made after every 10 epochs in the summary that is being generated.

4. Results and Discussion

Figure 4 shows at a high level a depiction of on how the abstractive summary generating decoder fetches words that are similar for any index that the neural network outputs. We were curious about how the construction and semantics of particular sentences affected the distribution in vector spaces. On inputting some simple sentences into the model which contained similar topics, but different semantic meanings projected into several different word-level constructions. We could potentially make some significant improvements by coming up with some way to reward semantic capture more effectively.

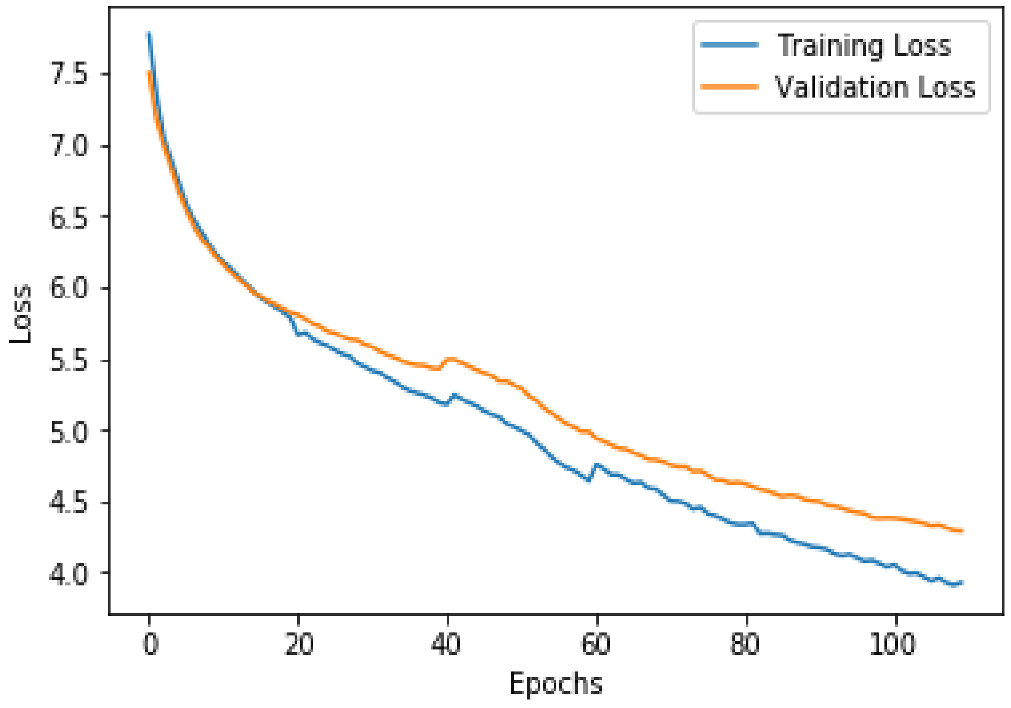


Figure 5: Training and Validation loss. The network was trained for 110 epochs with batches of size 32 each.

LSTMs and their ability to learn long term dependencies have been thoroughly studied, and problems in language when modelled solely as sequence problems come at the loss of understanding the semantics and syntax etc. Learning a good sentence vector representation and a good measure to evaluate their embeddings are very crucial for tackling problems such as summarization. Another interesting extension to our work is that of using a word model like that of Google Byte Pair Encoding or a much simpler model that leverages character level representations to create a mixed word-character vocabulary, as shown in Google's NMT system.

A model with plain global attention [3] and one more with greedy search decoding were also experimented on, but the former had a very large training time and did not converge in time enough o be documented in this report, and in the latter the generated outputs were not as close to the ground truth summaries.

5. Conclusion and Future Work :

We also have a very exciting extension in our pipeline, on using Transfer Learning to achieve Zero-Shot or One-Shot Summarization. The idea is to transfer the learning from the network trained for a particular task to another related task that offers more challenges like say scare data, or difficult to generalize etc. We have a plan to test French summary generation from French input texts by transferring the learning from English summarization and English to French translation learnt simultaneously. Previously, Google in one of its seminal papers established the idea of Zero Shot translation where mappings from one language to other was achieved without training on training samples of that problem. Our problem of French Summarization through Zero Shot Transfer Learning could provide the potential for tackling problems like Summarization in other alien languages without much data.

Acknowledgments

I would like to thank Prof. Matthew Stone for his valuable inputs and for the motivation he provided to all of us in aiming to take up challenging projects, and in spurring us to think along interesting research directions that need to be considered when thinking about the future of NLP, and general Machine Learning problems, and applications.

References

[1] Data : https://github.com/harvardnlp/sent-summary

[2] Rush, A., Chopra, S. & Weston, J. (2015) A Neural Attention Model for Sentence Summarization.
http://www.aclweb.org/anthology/D15-1044

[3] Luong et.al , Effective Approaches to Attention-based Neural Machine Translation,

[4] Bahdanau et.al Neural Machine Translation by jointly learning to Align and Translate

Appendix

The python code that was used for this study can be downloaded from Link. We have uploaded training weights that can be loaded onto the same model to reproduce our results on Summarization.