http://dx.doi.org/10.21172/ijiet.94.22

LSTM on different Word Embeddings

Minakshi Tomer[1]
[1]*MSIT, GGSIPU, New Delhi, India*

Abstract- Abstractive text summarization is an extremely challenging problem and can be solved by understanding contextual meaning of the text. In this paper a LSTM attention-based encoder-decoder architecture on the Gig word dataset is implemented and have demonstrated this by generating summaries of approximately 10 words, for a 30 - 40-word text. The novel idea in this paper is a different word embedding is being applied on the model. The results are evaluated using standard ROUGEscores and model using wordnet generate better results.
Keywords-LSTM,encoder-decoder,attention, abstractive, sequence to sequence, Natural Language Processing.

I. INTRODUCTION

Summarization is the process of decreasing the length of a text document while keeping the important information from the original document [1,2]. Technologies that can create comprehensible summaries often take into consideration the length, writing structure, and grammar of the text. In Natural Language Processing, there are mainly three approaches for summarization: The first method maintains the order of words of the input text and removes the least important words. The second method generates the summaries by only keeping words in the input text and not the word order. The third and the best approach, that generates human-like summaries and is the most arduous amongst the three, is abstractive summarization that produces summaries using any words from the vocabulary and not just from the input text [3][4]. With the help of Neural Network models, we can obtain accurate summaries primarily because of its knowledge of words and their analogy. It can be seen that employing an encoder-decoder model facilitates for abstractive summaries being produced at the decoder end because of the decoding segment selecting one word over the other which are analogous in some context. Sequence-to-Sequence models have been applied to achieve state-of-art results on most NLP tasks which affect text generation[5][6]. Use of robust LSTM and GRU based neural network layers has enabled the encoding long- term dependencies in input and target sequences.

Before the origin of neural networks to Natural language problems, Hidden Markov Models and chain models were utilized on these problems, but they had a significant fault: creating many wrong independent assumptions about the dependencies of tokens in a sequence[7]. LSTM and related neural network paradigms have the ability to encode and decode sequences without such assumptions and this is the principal reason for their qualitative performance in such problems.

II. MODEL

Decoding of words from indexes is achieved by Beam search decoder. The decoder outputs the most probable words at all times and generally does not give very good results. Today's state of the art approaches uses 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.

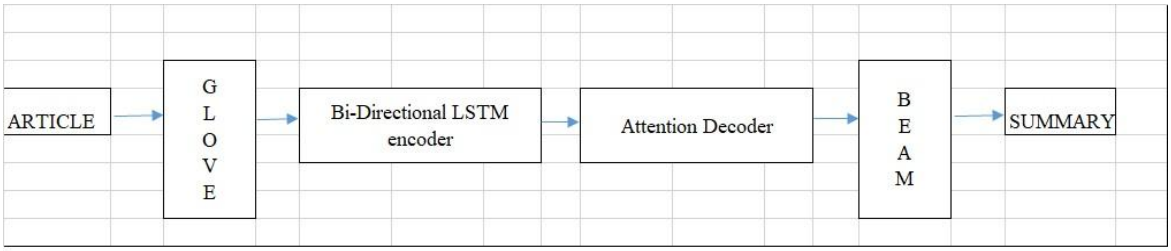

Fig1. Architecture of LSTM based model.

RNN:[8] Recurrent Neural Networks (RNN) are the only neural networks that have internal memory which makes it the most powerful algorithm out there for performing extremely memory intensive tasks.RNN's are able to retain essential things about the information they receive, which allows them to be very accurate in predicting the next

stage. In an RNN, the data circles over a loop. When a judgment is made, the present input and further what it has gained from the inputs it got before are taken into consideration.
RNN's suffered from 2 problems: -
Vanishing Gradient Problem - The weaker the gradient is, it gets more difficult for the network to update the weights and it takes more time to get the final result.
Exploding Gradient Problem - Large error gradients keep accumulating and result in extremely massive updates to neural network model weights while training. This produces the outcome of the model being unpredictable and incapable of learning from the training data.
To overcome the problems of the Recurrent Neural Network LSTM's were introduced.
LSTM: LSTMs [9][13] are intended to evade the long-term dependence dilemma. Retaining knowledge for extended intervals of time is effectively their default response. LSTMs too possess the chain-like composition, but the recurrent module has a distinct structure. Rather than having a singular neural network layer, there are four, communicating in a very specific behavior.LSTM is to estimate the conditional probability p ($y1$, . . ., $yT'|x1$, . . ., xT) where ($x1$, . . ., xT) is an input sequence and ($y1$, . . ., yT') is its corresponding output sequence whose length T' may differ. The LSTM computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence ($x1$, . . ., xT) given by the last hidden state of the LSTM, and then computing the probability of ($y1$, . . ., yT') with a standard LSTM-LM formulation whose initial hidden state is set to the representation v .
Unique Attention Model: The attention [13] 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. 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.

III. SYSTEM ARCHITECTURE

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.
Dataset:
The 50,000 description-summary pairs from the English Gigaword dataset with a maximum of 25 and 10 words each is being used. A vocabulary of 50000 most common words in this sample data set is taken and assigned Glove vector weights. In the remaining out of vocabulary (OOV) words [10], 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 pre-padded with zeros and output summaries are post-padded. The input to the network is concatenated sequence of description followed by summaries separated by <eos> label. The output of the network is the summary alone post-padded after a <eos> label to terminate the summary sequence.
There are three commonly used word embedding techniques:
Word2vec, wordnet and Glove.The performance of GloVe,word2vec and wordnet is evaluatedand compared in this paper.
GloVe:GloVe is an unsupervised learning algorithm[11] for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
GLOVE works similarly as Word2Vec. While you can observe preceding that Word2Vec is a "predictive" paradigm that predicts context provided word, GLOVE learns by creating a co-occurrence matrix (words X context) that essentially counts how often a word appears in context. Considering it's going to be a huge matrix, we factorize this matrix to achieve a lower-dimension representation.
The statistics of word occurrences in a corpus is the primitive cause of learning accessible to all unsupervised approaches for getting word representations. The objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.

IV. IMPLEMENTATION AND RESULTS

The model was trained on Google Collaboratory for 80 epochs and each epoch took a batch size of 32.
The output contains 3 sections:
• Original Summary: The original summary of the article from the dataset.
• DESC: Actual article or paragraph whose summary has to be generated.

• Generated Summary: Summary generated by our model.
Below are outputs of the test cases for reference:

```
Original Summary
hong kong shares open higher
DESC: hong kong share prices rose #.# percent on tuesday on selective buying , dealers said .
Generated Summary:
hong kong shares close #.# percent higher
```

Fig 2. Example 1

Original summary: Maoists bomb coca-cola plant in southern Nepal.

DESC: Maoist rebels in Nepal have bombed a Coca-Cola plant, the second attack on the factory in three months, police said Monday.

Generated summary: Maoists station rising stocks logging workers.

Fig 3.Example 2

Generated summaries for a few sample inputs including sample 1 shows very accurate summaries being generated. For stocks and inflation-related inputs, this model generated accurate summaries, and this was due to the predominance of samples talking about the corresponding in the training set. 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. This could potentially make some significant improvements by coming up with some way to reward semantic capture more effectively.

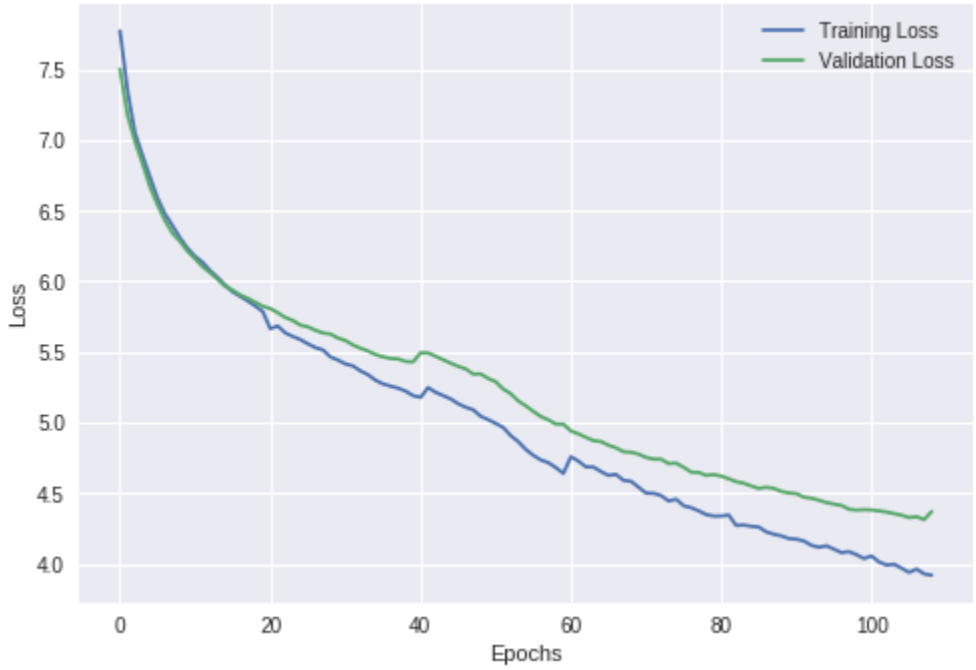

Fig 4. Training and Validation loss.

Training and validation loss during training this model is mentioned in figure 4. LSTMs and its capability to learn long-term dependencies have been studied thoroughly, and problems in language when modeled 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.

4.1 Evaluation Metric:

ROUGE [12] is a suite of evaluation metrics used for automatic text summarization. A summary is produced by a machine summarization system which is firstly compared against one or more hand-written summaries and then secondly assigned a score from 0 to 1. Rouge is categorized in multiple form: Rouge 1 is where there is unigram similarity occurs between gold standard summary and system summary. Rouge-2 is bigram similarity between gold standard summary and system summary. Rouge-L is the longest common sequence similarity between gold standard summary and system summary.

Table 1. Comparison of Rouge values using different word embeddings

Word embeddings	Rouge 1	Rouge 2	Rouge L
LSTM Glove	33.72	20.57	31.82
LSTM Word2vec	29.48	21.90	30.64
LSTM WordNet	36.08	24.82	33.17

As it is evident from the table, the WordNet embeddings give the highest ROUGE score instead of the Glove embeddings because the WordNet embeddings are hierarchical and symbolic whereas Glove is multidimensional.

V. CONCLUSION

This paper presented a neural attention-based model for abstractive summarization, based on recent developments in neural machine translation. In this probabilistic model is combined with a generation algorithm which produces accurate abstractive summaries. The dataset used is Gigaword and applied three different word embeddings.As the results shows wordnet based word embedding generate better results.In future the grammaticality of the summaries can be improved in a data-driven way, as well as scale this system to generate paragraph-level summaries. Both pose additional challenges in terms of efficient alignment and consistency in the generation.

VI. REFERENCES

[1] Ani Nenkova, K. M. (2012). A SURVEY OF TEXT SUMMARIZATION TECHNIQUES. In Mining Text Data (pp. 43-76). springer.
[2] Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. Proceedings of the 2016 Conference of the North American Chapter of the Association forComputational Linguistics: Human Language Technologies, 93-98.
[3] Jones, S. (2007). Automatic summarising: The state of the art. Information Processing and Management 43(6), 43, 1449-1481.
[4] Luhn, H. (1958). The Automatic Creation of Literature Abstracts. IBM jornal of Research and development, 159-165.
[5] Palomar, E. L. (2012). Text summarisation in progress: a literature review. artificial Intelligence Review 37, 1-41.
[6] Gambhir, M., and Gupta, V.: Recent automatic text summarization techniques: a survey, Artificial Intelligence Review 47(1), (2017) 1-66.
[7] J.Conroy, D. (2001). Text summarization via hidden markov models. proceedings of the Annual international ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 406-407.
[8] Rush, A. M., & Weston, S. C. (2015). A Neural Attention Model for Abstractive Sentence Summarization. arXiv:1509.00685v2 [cs.CL] 3 Sep 2015.
[9] Cheng,J.,Lapata,M.: Neural Summarization by Extracting Sentences and Words. arXiv:1603.07252v3 [cs.CL] 1 Jul 2016, (2016).
[10] See, A., & manning, p. J. (2017). Get To The Point: Summarization with Pointer-Generator Networks. arXiv:1704.04368 [cs.CL].
[11] Young, T., & al., D. H. (2018). Recent Trends in Deep Learning Based Natural language processing. arXiv:1708.02709v6 [cs.CL] 4 Aug 2018.
[12] Steinberger, J., & Je˘zek, K. (2007). EVALUATION MEASURES FOR TEXT SUMMARIZATION. Computing and Informatics, Vol. 28, 2009, 1001–1026, V 2009-Mar-2.
[13] Nallapati,R.,Zhou,B. and Santos,C.: Sequence-to-sequence rnns for text summarization. ICLR workshop abs/1602.06023., (2016).