An Abstructive Text Summarization Using Bidirectional LSTM

Shakeel Ahmed

Department of Computer Science
National University of Computer Emerging Sciences Islamabad
Shakeelsial05@gmail.com

Abstract

Text summarization is a technique of producing a short, concise and fluent summary of a text while preserving key information and overall meaning of that text. There are two different approaches used for text summarization. The first is extractive summarization in which we point out important sentences from the original text and extract it. The second is abstractive summarization, for which we are going to purpose methodology. Abstractive text summarization is the method of constructing summary sentences by merging facts from different source sentences and merging them into a shorter and concise representation while preserving all information of content and overall meaning. In abstractive text summarization, all the phrases and sentences are used to capture the meaning of the text document are entirely new and sentences and phrases used in the original text documents are rarely used. For that reason, the abstractive method is more complex as compared to extractive. However, these models have two drawbacks: they are unable to produce factual details accurately and they tend to repeat them. For overcoming these shortcomings, in this paper, we purposed a novel approach for abstractive summarization. We used a bi-directional LSTM based neural network architecture for abstractive level summarization of text. We are hopeful that our proposed methodology will outperform the current state of the art approaches for abstractive text summarization and overall not making the model complex.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

General Terms term1, term2

Keywords abstractive summarization, text summarization

1. Problem statement

- In abstractive text summarization, all the sentences and phrases which are used to capture the meaning of the input text are entirely new, and there are very little chances that the sentences of the original text will be used. For that reason, the abstractive methods for text summarization are more complex as compared to extractive text summarization. As the machine has to read over the text document and have to understand certain concepts and words to be important a preserve importing meaning. Then the machine fetches some cohesive phrasing of the relevant concepts. Abstractive text summarization is similar to how we being human to summarize, as often we summarize by paraphrasing. Due to the complexity of the abstractive summarization, the approach of summarization suffers from two big problems. Firstly, the summaries sometimes reproduce factual details inaccurately (e.g. Germany beat Argentina 3-2). This is very common for rare or out of vocabulary words such as 2-0. Secondly, the summaries sometimes repeat themselves(e.g. Germany beats Germany beats....). These problems are common for Neural networks in general. As in deep learning, it is hard to explain why the network exhibits such behavior. To overcome these problems we proposed our Bidirectional LSTM based model for abstractive text summarization.

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

2. Introduction

Text summarization is a method of producing a short and fluent summary consisting of a few sentences that can capture silent ideas of a text. This is an important task in the domain of natural language understanding. A concise and good summary of a document can help humans comprehend the text document better in very less time. My motivation for this project is from my own experience. As a student, we often faced with a large number of scientific papers and research articles that are very important for us and also belong to our interests, but we can't read them due to the length of the document or short time. So by implementing this project, I want to be able to get the summaries of main ideas of the research papers, without significant loss of content. Text summarization is a widely used and implemented algorithm but I want to use a different approach that can be applied to scientific papers and articles as well. The internet now a day has brought massive amounts of information to the fingertips of billions. If only we had time to read the information. Though we have been transformed by ready access to limitless data, we also find ourselves ensnared by information overload. For this reason, automatic text summarization, the task of automatically summarizing a piece of text to a shorter version, is now a day becoming increasingly vital. Because we have a large amount of information in the form of text, but we are not able to read all the text due to time shortage, so It is very useful to summarize the text so that we can get all the information in a short time, eliminating all the useless text. The automatic text summarization can be used in media monitoring, newsletters, internal document overflows, financial research, legal contract analysis, social media marketing, question answering bots, video scripting, books and literature and many more. There are broadly two approaches to automatic text summarization: extractive and abstractive. In extractive text summarization the passages are selected form the source text, then arrange them to form a summary. we can think of these ap-

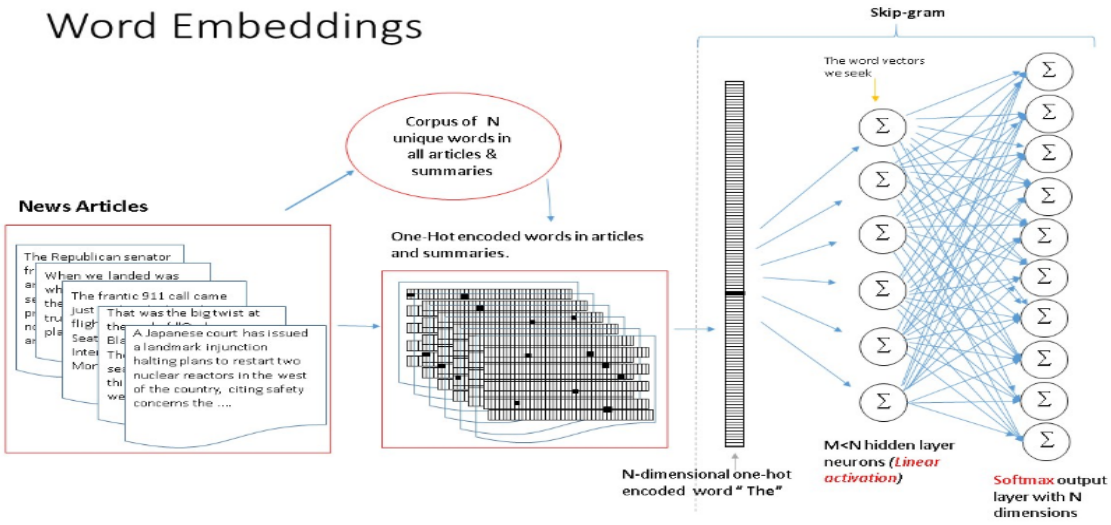


Figure 1. Word Embedding to SkipGram Model

proaches as a highlighter. In the abstractive approach, the natural language generation techniques are used to write the novel sentences by the same analogy, these approaches are like a pen. The most existing approaches to automatic text summarization are extractive because it is much easier to select text than to generate it from scratch. And as an extractive approach involves selecting and rearranging the whole sentences from the source text, it is guaranteed to produce a summary that is grammatical, fairly readable, and related to the source text. The models of these approaches are reasonably successful while applying to mid-length factual texts such as news articles and technical documents. on the other hand, the extractive approach is too restrictive to produce human-like summaries, especially of longer, more complex text. for example, we write a Wikipedia style plot synopsis of a novel says, Great Expectations, totally selecting and rearranging sentences from the book. This would be impossible. For one thing, great expectations are written in the first person whereas a synopsis should be in the third person. More importantly, condensing whole chapters of actions down to a sentence like Pip visits Miss Havisham and falls in love with her adopted daughter Estella requires powerful paraphrasing that is possible only in an abstractive framework. In short abstractive summarization may be difficult but it is essential.

But this approach to text summarization has two big drawbacks that we have to overcome. Firstly, the summaries sometimes do not reproduce factual detail correctly (e.g. Germany beat Argentina 3-2). This is especially common for rare and out of vocabulary words such as 2-0. Secondly, the summaries sometimes repeat the sentences again and again (e.g. Germany beat Germany beat....). The fact is, these problems mostly occur for RNNs in general. Because in deep learning, it is hard to explain why the network exhibits any particular behavior. The sequence to sequence with attention model makes it so difficult and hard to copy a word w from the input text. The network must have to generate that word after the information has passed through several layers of computations, including mapping w to its word embedding. If w is a rare word that not appeared frequently during the training and has a poor embedding (then it will be clustered with totally unrelated words), then w is, from the perspective of the network, indistinguishable from many many other words, thus it is hard to reproduce. Even if w has

Original Article: McDonald's says..... The company says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates, which it said was used to keep the chicken moist, in favor of vegetable starch. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning. Jessica Foust, director of culinary innovation at McDonald's, said the changes were made because customers said they want 'simple, clean ingredients' they are familiar with..... And *Panera Bread has said* it plans to purge artificial colors, flavors and preservatives from its food by 2016.....

Extractive Approach: The company says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates, which it said was used to keep the chicken moist, in favor of vegetable starch. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning.

Abstractive Approach: McDonald's says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week. The company says the changes were made because customers said they want 'simple, clean ingredients' they are familiar with. *McDonald's said* it plans to purge artificial colors, flavors and preservatives from its food by 2016.

Unified Approach: McDonald's says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning.

Figure 2. Abstractive and Extractive Summarization Example

a good word embedding, the network may still have difficulty reproducing the word. For example, RNN summarization systems mostly replace a name with another name(e.g. Islamabad to Rawalpindi). This is because the word embedding for e.g. female names or Indian cities tends to cluster together, which may confuse when attempting to reconstruct the original word. This is an unnecessarily difficult way to perform a simple operation copying, that is a fundamental operation in summarization. The

second problem is the repetition of words, It may be caused by the decoder's over-reliance on the decoder input(i.e previous summary word), rather than storing longer-term information in the decoder state. This is observed that single repeated words commonly trigger an endless repetitive cycle (e.g. Germany beat Germany beat....). Recent work in text summarization is dominated by neural networks based methods, namely recurrent neural networks (RNN). Most of which used encoder-decoder architecture, in which RNN models based encoder-decoder has used. It achieved state of the art results on short text summarization[12]. They proved that the most efficient method plays the role of encoder and decoder usually, as an encoder that transforms sequential data into vectors, and as a decoder that transforms the encoded vector into sequential output [23]. For a performance increase, they also used attention as an extra input to the decoder. although encoder-decoder methods can generate sentences with correct grammar, they still have difficulties in generating long sentences with rich semantics. These sentences are also robotic and unable to get to the humans level. Since generating longer text summarization requires a higher level of abstraction while also reducing and avoiding the repetitions, It is more challenging yet more useful [22]. The previous text summarization approaches which are based on neural networks are categorized into two categories: extractive and abstractive. Extractive summarization aims to select and concatenate the sentences that retain important information from a document to generate its summary [23][1]. Abstractive text summarization techniques build semantic representations internally of the original text and then create a summary closer to a human-generated one, which needs a deeper concept of text. Abstractive Text summarization is still challenging and still quite weak, most of the previous work has focused on extractive summarization [1] To the best of our knowledge, all the current abstractive models use a unidirectional decoder, usually an RNN variant. In making predictions, a unidirectional RNN needs to encode the previous predictions are incorrect, but it may include some noise, which undermines the quality of the subsequent predictions. Typically in unidirectional models, a general way to predict the next token is to maximize the log probability of given the encoder output and the decoder output generated at the last steps. since unidirectional models only reason about the past, they are still limited to retain the future context. This limitation makes the unidirectional models to suffer on its own by not using the past and the future dependencies while prediction. For this, we don't have used the unidirectional model because of its limitations. Instead, we purposed a bidirectional LSTM model for abstractive level text summarization.

3. Related work

[20] was the first which applied neural networks to abstractive summarization, achieving state of the art performance on DUC-2004 and Gigaworld, two-sentence level summarization data sets. They used attention mechanism, have been augmented by recent decoders [6], abstract meaning representations [24], hierarchical networks [15],variational autoencoders [14],and direct optimization of the performance matric [18],further improving performance on these data sets. But large scale datasets for summarizations of longer text are rare. [15] adapted the deep mind question-answering dataset [9] for summarization, resulting in CNN/daily mail dataset and provided the first abstractive baseline. The same author than published an extractive neural approach [16], which uses RNN for selecting sentences, and got that it is outperformed abstractive summarization comparing to ROUGE metric. these are the only two results published on the full dataset. Before modern neural networks, abstractive summarization receives less attention as compared to extractive summarization. but [10] explored cutting unimportant parts of sentences to generate summaries. and [5] explored sentence fusion using dependency trees. The pointer network [26] is a sequence-to-sequence model that uses the soft attention distribution of [2] to produce an output sequence consisting of elements from the input sequence. The pointer network has been used to create hybrid approaches for NMT [8], language modeling [13], and summarization ([7][8] [14][15][28]).// Our approach is close to the Forced-Attention Sentence Compression model of[14] and the CopyNet model of [7], with some small differences: (i) We calculate an explicit switch probability pgen, whereas Gu et al. induce competition through a shared softmax function. (ii) We recycle the attention distribution to serve as the copy distribution, but Gu et al. use two separate distributions. (iii) When a the word appears multiple times in the source text, we sum probability mass from all corresponding parts of the attention distribution, whereas Miao and Blunsom do not. Our reasoning is that (i) calculating an explicit pgen usefully enables us to raise or lower the probability of all generated words or all copy words at once, rather than individually, (ii) the two distributions serve such similar purposes that we find our simpler approach suffices, and (iii) we observe that the pointer mechanism often copies a word while attending to multiple occurrences of it in the source text. Our approach is considerably different from that of [8] and [15]. Those works train their pointer components to activate only for out-of-vocabulary words or named entities (whereas we allow our model to freely learn when to use the pointer), and they do not mix the probabilities from the copy distribution and vocabulary distribution. We believe the mixture approach described here is better for abstractive summarization – in section 6 we show that the copy mechanism is vital for accurately reproducing rare but in-vocabulary words, and in section 7.2 we observe that the mixture model enables the language model and copy mechanism to work together to perform abstractive copying. Originating from Statistical Machine Translation [11], coverage was adapted for NMT by [25] and [**?**], who both use a GRU to update the coverage vector each step. We find that a simpler approach – summing the attention distributions to obtain the coverage vector – suffices. In this respect our approach is similar to [27], who apply a coverage-like method to image captioning, and [3], who also incorporate a coverage mechanism (which they call 'distraction') into neural summarization of longer text. Temporal attention is a related technique that has been applied to NMT [21] and summarization [15]. In this approach, each attention distribution is divided by the sum of the previous, which effectively dampens repeated attention. We tried this method but found it too destructive, distorting the signal from the attention mechanism and reducing performance. We hypothesize that an early intervention a method such as coverage is preferable to a post hoc method such as temporal attention – it is better to inform the attention mechanism to help it makes better decisions than to override its decisions altogether. This theory is supported by the large boost that coverage gives our ROUGE scores (see Table 1), compared to the smaller boost given by temporal attention for the same task [15].

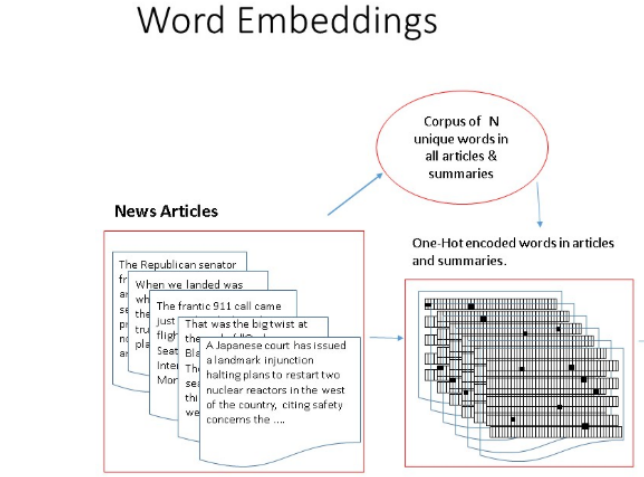
4. Purposed approach

In this work, we are going to use bidirectional lstm for abstractive level text summarization. For that firstly we have to do world embedding.

4.1 Word Embedding

- Word embedding is a type of word representation that allows words with similar meaning to have a similar representation.

  They are a distributed representation for the text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems. [ht] embeding2.png



. Word Embedding

A word embedding is a learned representation for text where words that have the same meaning have a similar representation.

It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

One of the benefits of using dense and low-dimensional vectors is computational: the majority of neural network toolkits do not play well with very high-dimensional, sparse vectors. ... The main benefit of the dense representations is generalization power: if we believe some features may provide similar clues, it is worthwhile to provide a representation that can capture these similarities.

Word embeddings are a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

The key to the approach is the idea of using a densely distributed representation for each word.

Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one-hot encoding.

associate with each word in the vocabulary a distributed word feature vector ... The feature vector represents different aspects of the word: each word is associated with a point in a vector space. The number of features ... is much smaller than the size of the vocabulary.

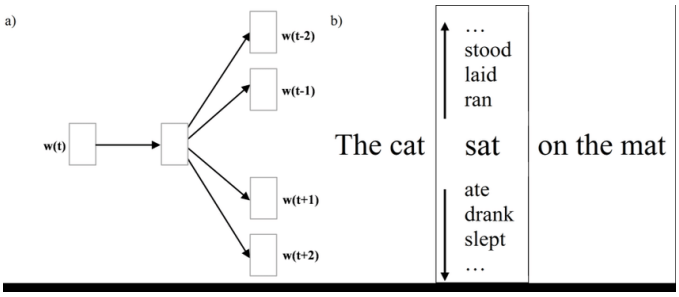


Figure 3. Skip Gram Model

The distributed representation is learned based on the usage of words. This allows words that are used in similar ways to result in having similar representations, naturally capturing their meaning. This can be contrasted with the crisp but fragile representation in a bag of words model where, unless explicitly managed, different words have different representations, regardless of how they are used.

There is the deeper linguistic theory behind the approach, namely the "distributional hypothesis" by Zellig Harris that could be summarized as: words that have a similar context will have similar meanings. Word embedding is a popular representation of document vocabulary capable of capturing the context of a word in a document. It is a combined name for a set of language modeling and feature learning method used in natural language processing. In word embedding, the words and phrases from vocabulary are mapped to vectors of real numbers.

This notion of letting the usage of the word define its meaning can be summarized by an oft-repeated quip by John Firth: "You shall know a word by the company it keeps!" There are many algorithms in the world2vec library for the encoder input sequence. For purposed work, we are going to use the skip-gram algorithm for the encoder input sequence.

4.2 Skip Gram Model

- As the vocabulary of any language is large and cannot be labeled by humans and hence we require unsupervised learning techniques that can learn the context of any word on its own. Skip-gram is one of the unsupervised learning techniques used to find the most related words for a given word. Skip-gram is used to predict the context word for a given target word. It's the reverse of the CBOW algorithm. Here, the target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult. The word sat will be given and we'll try to predict words cat, mat at position -1 and 3 respectively given sat is at position 0. We do not predict common or stop words such as the .

Just like we discussed in the CBOW model, we need to model this Skip-gram architecture now as a deep learning classification model such that we take in the target word as our input and try to predict the context words. This

becomes slightly complex since we have multiple words in our context. We simplify this further by breaking down each (target, context words) pair into (target, context) pairs such that each context consists of only one word. Hence our dataset from earlier gets transformed into pairs like (brown, quick), (brown, fox), (quick, the), (quick, brown) and so on. But how to supervise or train the model to know what is contextual and what is not.

For this, we feed our skip-gram model pairs of (X, Y) where X is our input and Y is our label. We do this by using [(target, context), 1] pairs as positive input samples where the target is our word of interest and context is a context word occurring near the target word and the positive label 1 indicates this is a contextually relevant pair. We also feed in [(target, random), 0] pairs as negative input samples where the target is again our word of interest but random is just a randomly selected word from our vocabulary which has no context or association with our target word. Hence the negative label 0indicates this is a contextually irrelevant pair. We do this so that the model can then learn which pairs of words are contextually relevant and which are not and generate similar embedding for semantically similar words.

4.3 Encoder-Decoder LSTM

- The Encoder-Decoder LSTM is a recurrent neural network designed to address sequence-to-sequence problems, sometimes called seq2seq.

  Sequence-to-sequence prediction problems are challenging because the number of items in the input and output sequences can vary. For example, text translation and learning to execute programs are examples of seq2seq problems. Sequence prediction often involves forecasting the next value in a real-valued sequence or outputting a class label for an input sequence.

  This is often framed as a sequence of one input time step to one output time step (e.g. one-to-one) or multiple-input time steps to one output time step (many-to-one) type sequence prediction problem.

  There is a more challenging type of sequence prediction problem that takes a sequence as input and requires a sequence prediction as output. These are called sequence-to-sequence prediction problems, or seq2seq for short.

  One modeling concern that makes these problems challenging is that the length of the input and output sequences may vary. Given that there are multiple input time steps and multiple-output time steps, this form of the problem is referred to as many-to-many type sequence prediction problem. One approach to seq2seq prediction problems that has proven very effective is called the Encoder-Decoder LSTM.

  This architecture is comprised of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. The use of the models in concert gives the architecture its name of Encoder-Decoder LSTM designed specifically for seq2seq problems.

  RNN Encoder-Decoder consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence.

  The Encoder-Decoder LSTM was developed for natural language processing problems where it demonstrated state-of-the-art performance, specifically in the area of text translation called statistical machine translation.

  The innovation of this architecture is the use of a fixed-sized internal representation in the heart of the model that input sequences are read to and output sequences are read from. For this reason, the method may be referred to as sequence embedding.

encoder lstm.png

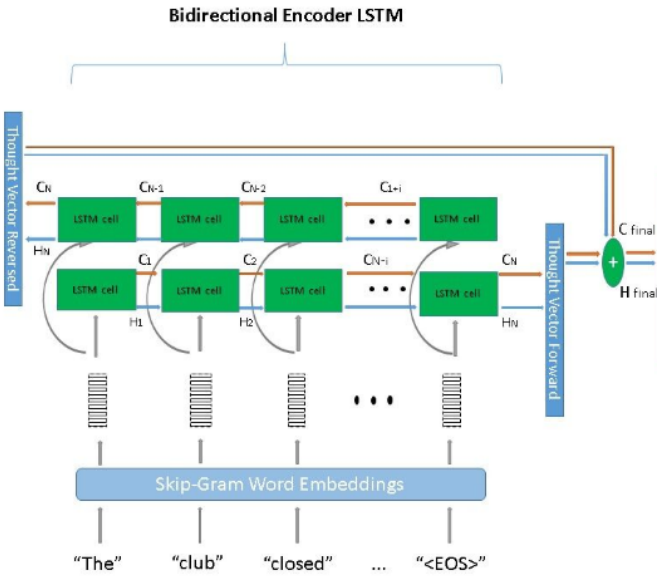


Figure 4. Bidirectional Encoder LSTM

- In one of the first applications of the architecture to English-to-French translation, the internal representation of the encoded English phrases was visualized. The plots revealed a qualitatively meaningful learned structure of the phrases harnessed for the translation task.

  The proposed RNN Encoder-Decoder naturally generates a continuous-space representation of a phrase. From the visualization, it is clear that the RNN Encoder-Decoder captures both semantic and syntactic structures of the phrases.

  On the task of translation, the model was found to be more effective when the input sequence was reversed. Further, the model was shown to be effective even on very long input sequences.

  We were able to do well on long sentences because we reversed the order of words in the source sentence but not the target sentences in the training and test set. By doing so, we introduced many short term. dependencies that made the optimization problem much simpler. . . . The simple trick of reversing the words in the source sentence is one of the key technical contributions of this work. This approach has also been used with image inputs where a Convolutional Neural Network is used as a feature extractor on input images, which is then read by a decoder LSTM.

  . . . we propose to follow this elegant recipe, replacing the encoder RNN by a deep convolution neural network (CNN). it is natural to use a CNN as an image encoder", by first pretraining it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences.

decoder lstm.png

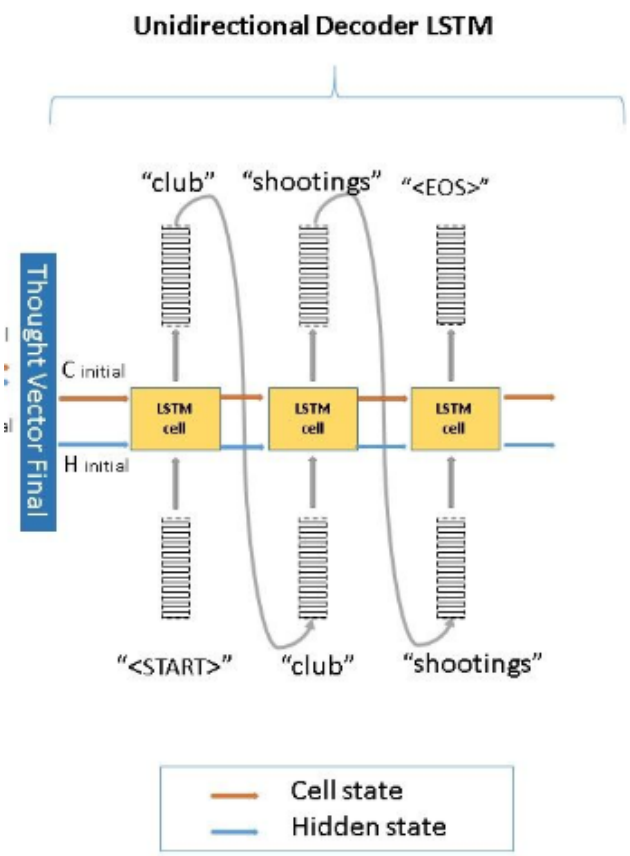


Figure 5. Uni-directional Decoder LSTM

layer.png

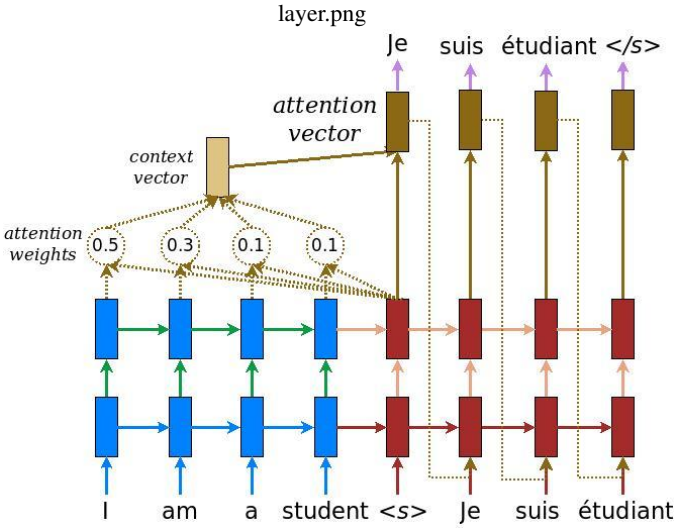


Figure 6. Working of Attention

4.4 Attention Layer

- An attention layer between the encoder and decoder over the source sequence's hidden states. as the skipgram embedding and the one-hot vector sizes aren't the same, pca over embedding to allow multiplication with one-hot vectors to get attention weights and vectors. final prediction of output word in decoder sequence is done by the attention layer. It helps allow the decoder individual encoder state information. Attention is simply a vector, often the outputs of dense layer using softmax function. Before Attention mechanism, translation relies on reading a complete sentence and compress all information into a fixed-length vector, as you can image, a sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc.

  However, attention partially fixes this problem. It allows machine translator to look over all the information the original sentence holds, then generate the proper word according to current word it works on and the context. It can even allow translator to zoom in or out (focus on local or global features). Attention is not mysterious or complex. It is just an interface formulated by parameters and delicate math. You could plug it anywhere you find it suitable, and potentially, the result may be enhanced.

The core of Probabilistic Language Model is to assign a probability to a sentence by Markov Assumption. Due to the nature of sentences that consist of different numbers of words, RNN is naturally introduced to model the conditional probability among words.

Translation often requires arbitrary input length and output length, to deal with the deficits above, the encoder-decoder model is adopted and basic RNN cell is changed to GRU or LSTM cell, hyperbolic tangent activation is replaced by ReLU. We use GRU cells here. Embedding layer maps discrete words into dense vectors for computational efficiency. Then embedded word vectors are fed into the encoder, aka GRU cells sequentially. What happened during encoding? Information flows from left to right and each word vector is learned according to the not only current input but also all previous words. When the sentence is completely read, the encoder generates an output and a hidden state at timestep 4 for further processing. For the encoding part, the decoder (GRUs as well) grabs the hidden state from the encoder, trained by teacher forcing (a mode that previous cell's output as current input), then generate translation words sequentially. Similar to the basic encoder-decoder architecture, this fancy mechanism plug a context vector into the gap between encoder and decoder. According to the schematic above, blue represents encoder and red represents decoder; and we could see that context vector takes all cells' outputs as input to compute the probability distribution of source language words for each single word decoder wants to generate. By utilizing this mechanism, it is possible for the decoder to capture somewhat global information rather than solely to infer based on one hidden state. And to build a context vector is fairly simple. For a fixed target word, first, we loop over all encoders' states to compare target and source states to generate scores for each state in encoders. Then we could use softmax to normalize all scores, which generates the probability distribution conditioned on target states. At last, the weights are introduced to make the context vector easy to train.

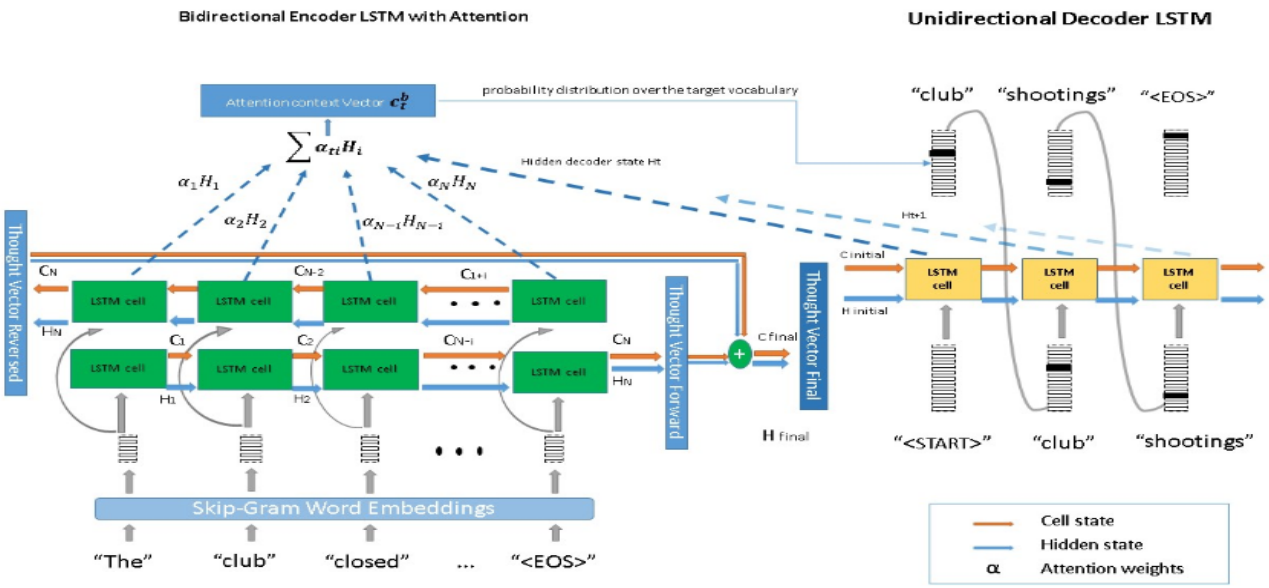


Figure 7. Encoder Decoder LSTM with Attention

5. Evaluation and Experiments

5.1 Datasets

- In this work, we are using CNN/Daily Mail dataset to evaluate our proposed model. CNN/Daily Mail dataset was originally built by [19] for question answering task and then re-used for extractive [4][16][1] and abstractive text summarization tasks [15][17][22]. In the joint CNN/Daily Mail dataset, there are 286,726 for training, 13,368 for validation and 11,490 for testing. The source document in the training has 781 words spanning 29.74 sentences on average and summaries consist of 56 words and 3.75 sentences.

5.2 Baselines

- There are many approaches and models that are recently proposed. We choose only those what are compareable to our work. We are using 4 baselines.

  RL:a reinforced abstractive summarization model, Pointer generator based network [22],SummaRuNNer-abs[16] and words-lvt2k [15] will be used in this work as abstractive baselines.

5.3 Experiment Settings

- We are aiming to learn from scratch on the CNN/Daily Mail datasets with a dimension of 128. The model will have 128 hidden units. The vocabulary size will be 30k or 25k depending upon memory constraints. We are aiming to use RMsprop optimizer to train our model with a batch size of 50, the learning rate will be 0.005 and we can also use regularization depending upon final results. For implementation and training, we are aiming to use google collab's GPU. The results will be shown in the final report.

5.4 Experimental Results

- ROUGE toolkit and pyrouge python package will be used to evaluate the results and performance of our purpose model. With time in the head, we are aiming to make our model complex for better performance. ROUGHE-1 ROUGHE-2 AND ROUGHE-L scores will be reported. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. We are aiming to use ROUGHE for evaluation of our model as it is being used mostly in all text summarization especially abstractive summarization and machine translation tasks.

References

[1] K. Al-Sabahi, Z. Zuping, and M. Nadher. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access*, 6:24205–24212, 2018.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Q. Chen, X.-D. Zhu, Z.-H. Ling, S. Wei, and H. Jiang. Distraction-based neural networks for modeling document. In *IJCAI*, pages 2754–2760, 2016.

[4] J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*, 2016.

[5] J. C. K. Cheung and G. Penn. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 775–786, 2014.

[6] S. Chopra, M. Auli, and A. M. Rush. 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*, pages 93–98, 2016.

[7] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[8] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*, 2016.

[9] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

[10] H. Jing. Sentence reduction for automatic text summarization. In *Sixth Applied Natural Language Processing Conference*, pages 310–315, 2000.

[11] P. Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[12] S. Li, L. Ye, W. Zhao, H. Yan, B. Yang, D. Liu, W. Li, H. Ade, and J. Hou. A wide band gap polymer with a deep highest occupied molecular orbital level enables 14.2% efficiency in polymer solar cells. *Journal of the American Chemical Society*, 140(23):7159–7167, 2018.

[13] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[14] Y. Miao and P. Blunsom. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*, 2016.

[15] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[16] R. Nallapati, F. Zhai, and B. Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[18] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[19] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[20] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[21] B. Sankaran, H. Mi, Y. Al-Onaizan, and A. Ittycheriah. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927*, 2016.

[22] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[23] Q. Sun, S. Lee, and D. Batra. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6961–6969, 2017.

[24] S. Takase, J. Suzuki, N. Okazaki, T. Hirao, and M. Nagata. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1054–1059, 2016.

[25] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*, 2016.

[26] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[28] W. Zeng, W. Luo, S. Fidler, and R. Urtasun. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*, 2016.