The Pennsylvania State University

The Graduate School

ABSTRACTIVE TEXT SUMMARIZATION BY HIGHLIGHTING

A Thesis in

Electrical Engineering

by

Rajeev Bhatt Ambati

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2019

The thesis of Rajeev Bhatt Ambati was reviewed and approved* by the following:

David Miller
Professor of Electrical Engineering
Thesis Advisor

Prasenjit Mitra
Professor of Information Sciences and Technology,
Associate Dean for Research.

Victor Pasko
Professor of Electrical Engineering,
Coordinator of the Graduate Program.

Kultegin Aydin
Professor of Electrical Engineering,
Head of the Department of Electrical Engineering.

*Signatures are on file in the Graduate School

ABSTRACT

Traditional sequence-to-sequence (seq2seq) models and other variations of attention mechanism such as hierarchical attention have been applied to the text summarization problem. Though there is a hierarchy in the way humans use language by forming paragraphs from sentences and sentences from words, hierarchical models have failed to perform better than their traditional seq2seq counterparts. This effect is mainly because either the hierarchical attention mechanisms are too sparse using hard attention; selecting the maximally aligned sentence or noisy using soft-attention; assigning an alignment score to each sentence/word and using a weighted sum. Long Short-Term Memory (LSTM) is the backbone of such seq2seq models, and the architectures are primarily inspired by the machine translation literature [1] [2] [3]. In a typical text summarization dataset [4] which consists documents that are 800 tokens in length, capturing long-term dependencies is very important, for example, if the last sentence can be grouped with the first sentence of a document. LSTMs often fail to capture long term dependencies while modeling long sequences. To address these issues, we have made the following contributions through this work:

- Adapted Neural Semantic Encoders (NSE) to text summarization by making the following changes:
    - Improved its attention mechanism by using a feed-forward neural network in place of simple dot-product attention.
    - Improved the compose function using an LSTM so that it can remember what is previously composed and maintain novelty.
    - Added a copy mechanism for facilitating the usage of document words in the summary. This helps the model in maintaining factual consistency with the document and also in the usage of words that are not present in our limited size vocabulary.

- Proposed a novel hierarchical NSE that performs better than previous abstractive summarization models of the same complexity and is also faster to train.
    - Separate memories are used for each sentence to enrich the word representation.
    - A shared document memory is used to enrich the sentence representation and also to retrieve the highlights of the document.
- Improved the best abstractive summarizer (supervised) by 1 ROUGE point.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am grateful to CHOT (The Center for Health Organization Transformation) for funding this project and my advisor Dr. Prasenjit Mitra for providing me this opportunity and also valuable guidance throughout the project. I thank all my family members for their fantastic support for two years of M.S program. I appreciate my committee members, Dr. David Miller, and Dr. Victor Pasko for their kind cooperation and time.

DISCLAIMER: Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the CHOT (The Center for Health Organization Transformation).

Chapter 1

Introduction

In the age of the Internet, we are bombarded with a lot of information, which we cannot possibly go through. The Washington Post alone publishes 500 news articles per day [5]. The necessity to process large amounts of text has provided a strong impetus to develop automatic text summarization systems. Text summarization is the problem of condensing large documents into short and readable summaries. It can be used for both single and multi-document summarization [6] [3] [7]. The text summarization task is broadly classified into two categories: extractive and abstractive summarization [1]. Extractive approaches select sentences from a given document and groups them to form concise summaries, and therefore, classification models are mostly used to solve this task. By contrast, abstractive approaches primarily capture the semantics of input documents and create summaries containing rephrased key content. The former task falls under the classification paradigm, and the latter belongs to the generative modeling paradigm, and therefore it is a much harder problem to solve. Figure-1 shows an example demonstrating the difference between both types of summarization.

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: 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: 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.

Figure **1**: This example [17] shows the difference between extractive and abstractive summarization. The extractive summary is made of text directly copied from the article (highlighted in green). On the other hand, the abstractive summary has a combination of different words - "McDonald's says" instead of "company says" (highlighted in blue). However, abstractive summarization is prone to errors – "Panera Bread" is conflated with "McDonald's" (highlighted in red) because both the nouns related to restaurants are very close in the word-vector space. Since it is a natural language generation task, an abstractive summarizer outputs a word vector that is close to both the word vectors of McDonald's and Panera Bread.

Background

In this section, we will introduce the basic concepts in sequence modeling that will come handy in discussing the drawbacks of previously proposed methods for text summarization and finally, state the motivation for our work in the next section.

Recurrent Neural Network

Sequence modeling is a general paradigm for studying sequential data. Language modeling if one of the most exciting topics that use sequence labeling such as language translation in which understanding the meaning of each word and the relationship between the words is quite essential. Plain feedforward neural networks are not good at adapting to varying length input and output. The inputs and outputs are assumed to be independent of each other, which is not valid for sequential data, for example, in a video of a car race, each frame is dependent on the previous frame. The idea behind an RNN is to make use of this sequential information. Recurrent neural networks have two sources of input: the present and the recent past; this is stored in the form of the hidden state. At each time-step, the hidden state is updated continuously as per equation (1).

$$h_t = f_W(h_{t-1}, x_t) \tag{1}$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{2}$$

$$y_t = W_{hy}h_t \tag{3}$$

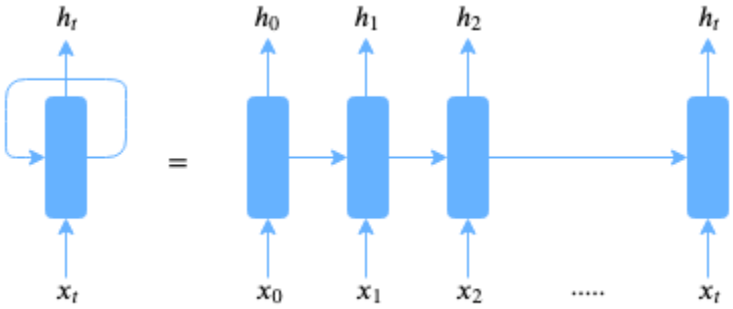

Figure **2**: This figure depicts an unrolled RNN. The RNN looks at the input x_t and updates its hidden state h_t sequentially for t times (shown on the left); this is same as the unrolled RNN which takes the input at each time-step, updates the hidden state and passes it to the next RNN cell (shown on the right).

The input features are extracted by projecting x_t, the input at time step t, using the weight matrix W_{xh}. Similarly, the hidden state features are extracted by projecting h_{t-1}, the previous hidden state, using the hidden-state-to-hidden-state matrix W_{hh}; this is known as the transition matrix that is similar to a Markov Chain. These features are added and passed through a tanh activation function to obtain the new state h_t as shown in equation (2). Finally, the output y_t is calculated by multiplying a state-output transformation matrix W_{hy} with the hidden state h_t using equation (3). The above weight matrices are filters that determine how much importance to accord to both input and past hidden state. These parameters are learned by minimizing an objective function using backpropagation through time in an unrolled RNN shown in Figure-2. But the problem of vanishing or exploding gradients makes it hard to train an RNN. Equation (4), (5) below show the gradient accumulation after backpropagating from time step k to t where, $\beta_{W_{hh}}, \beta_h$ denote the upper bounds of the norms; this can become very small or very large quickly [18].

$$\left\|\frac{\partial h_t}{\partial h_{t-1}}\right\| \leq \|W_{hh}^T\| \|\text{diag}[f'(h_{t-1})]\| \leq \beta_{W_{hh}} \beta_h \tag{4}$$

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| = \left\|\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}\right\| \leq \left(\leq \beta_{W_{hh}} \beta_h\right)^{t-k} \tag{5}$$

It means that inputs that are time steps far away are not taken into consideration when training to predict the next word and results in a loss of memory. As we will see in the next section, Long Short-Term Memory uses a gating mechanism to overcome this problem.

Long Short-Term Memory

Long Short-Term Memory (LSTM) regulates the flow of information using a gated mechanism. Information can be stored in, written to, or read from the cell, much like data in a computer's memory. The cell decides as to what to store, erase, write, and communicate to the next cell. Unlike the digital storage on computers, these gates are analog, implemented with element-wise multiplication by sigmoid, which are all in range 0-1. Since they are differentiable, it facilitates the backpropagation.

$$f_t = \sigma\left(W_f^T[h_{t-1}, x_t] + b_f\right) \quad (6)$$

$$i_t = \sigma\left(W_i^T[h_{t-1}, x_t] + b_i\right) \quad (7)$$

$$\tilde{C}_t = \tanh(W_C^T[h_{t-1}, x_t] + b_C) \quad (8)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (9)$$

$$o_t = \sigma(W_o^T[h_{t-1}, x_t] + b_o) \quad (10)$$

$$h_t = o_t \odot \tanh(C_t) \quad (11)$$

First, the LSTM decides what information to throw away from the cell state; this decision is made by the "forget gate" using equation (6). It looks at the previous hidden state h_{t-1} and current input x_t, and outputs a number between 0 and 1 for each dimension in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

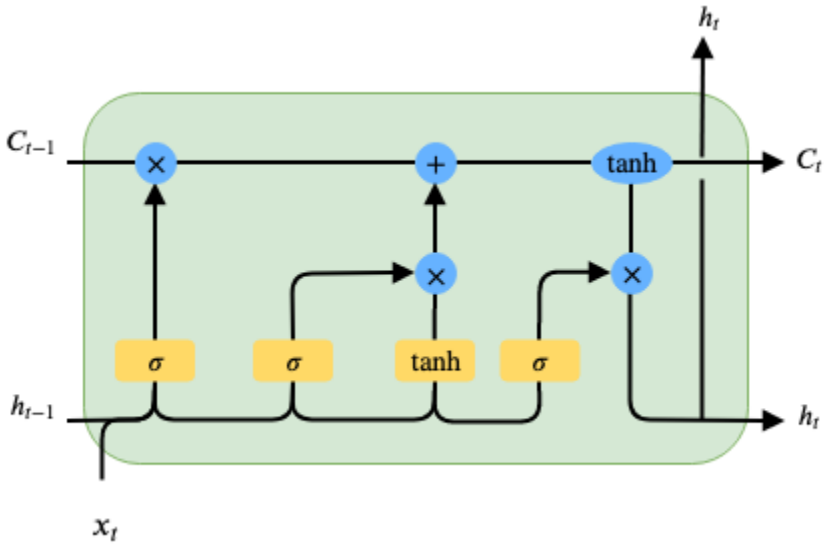

Figure **3**: This is a single cell of an LSTM[1], composed of input, forget and output gates that are computed using the previous hidden state h_{t-1} and current input x_t. New cell state is computed by erasing the previous cell state as per forget gate f_t and updating as per input gate i_t and candidate cell state $\tilde{C}_t$. Finally, the output h_t is sent to the next cell by the output gate.

The next step is to decide what is going to be stored in the cell state. Equation (7) describes a sigmoid layer called the "input gate" that determines the values to be updated. Next, new candidate values $\tilde{C}_t$ are created using a tanh layer as per equation (8). Now, the cell state is updated as shown in equation (9); the old cell state C_{t-1} is multiplied by f_t, thereby erasing the information to forget and added with the new candidate cell multiplied by i_t to scale the update as decided by the input gate. The final step is to compute the output based on the cell state. As usually, a sigmoid layer is used to determine what parts of the cell state is going to be used for the output as computed in equation (10). Then, the output gate is multiplied with a tanh version of cell state; this is to push the values between -1 and 1 using equation (11). Keep in mind that multiplying two sigmoid layers squishes the values even further, which might lead to vanishing gradient problem; therefore, the use of tanh might negate the small gradient from sigmoid.

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Word Representation

We cannot feed words as input to train the language models and need vectors to represent these words. Using one-hot representations (symbols) as lookups to each word is quite inefficient, for example, a vocabulary of 50,000 words need 50,000-dimensional vectors to represent each word in the vocabulary, and it is directly proportional with the size of the vocabulary. Therefore, learning distributed word vector representations has been one of the leading research areas since the beginning of NLP. Two main theories behind word representation learning can be classified as vector space models and word embeddings.

Vector Space Models

One of the most prominent methodologies for word representation learning is based on Vector Space Models (VSM) [19]. The earliest VSM applied in NLP first represented a document as a vector whose dimensions were the whole vocabulary. Each of these dimensions is assigned a weight based on the corresponding word frequency within the entire document. Primarily, the weight computation metrics based on frequencies or normalized frequencies were used. This methodology has been successfully adapted to various NLP applications such as information retrieval, text classification, or sentiment analysis, etc. This document-based VSM was extended to represent words in which the word-based vector is constructed based on normalized frequencies of the co-occurring words in a corpus. The main idea behind VSM is that words that share similar meanings should be close in the vector space. One of the main drawbacks of conventional VSM approaches is the high dimensionality of resulted word vectors. Since the dimensions represent the vocabulary, this can reach hundreds of millions depending on the underlying corpus. To reduce the representation size, various dimension reduction techniques such as Singular Value Decomposition (SVD), Latent Semantic Analysis (LSA) were used on top of the document matrices in which rows correspond to word count and columns correspond to different documents in the corpora. Another

way is to directly use neural networks to learn these low-dimensional distributed word vector representations.

Word Embeddings

Learning low-dimensional word representations from text data can be achieved by using neural networks models known as *word embeddings*. Mikilov et al. [20] proposed the Word2Vec model based on optimizing an efficient but straightforward architecture that exhibits interesting semantic properties. They proposed two different but related word2vec models: Continuous Bag-Of-Words (CBOW) and Skip-gram. The first model, CBOW, architecture is based on a feedforward neural network language model whose aims is to predict the current word using its surrounding context, by minimizing the log-likelihood predicting the target word given context words:

$$E = -\log p\left(\vec{w}_t \middle| \vec{W}_t\right)$$

Where, w_t is the target word and $W_t = w_{t-n}, \dots, w_t, \dots, w_{t+n}$ represents the sequence of words in the context. The resulting word vector space is good at encoding word analogies, for example, the analogy "king is to queen as man is to woman" is encoded in the vector space by the vector equation $king - queen = man - woman$. Another prominent word embedding architecture GloVe [21] combines global matrix factorization and local context methods through a bilinear regression model.

GloVe (Global Vectors for Word Representation)

Currently, both the families discussed above suffer significant drawbacks. While methods like LSA efficiently leverage statistical information of a corpus, they do relatively poorly in word analogy tasks, indicating a sub-optimal vector space structure. Methods like Skip-gram may do better on analogy task, but they poorly utilize the statistics of the corpus since they train on a separate local context window instead of global co-occurrence counts. Let X denote the matrix of

word-word co-occurrence counts, whose entries $X_{ij} = \sum_k X_{ik}$ be the number of times any word appears in the context of word i. Similarly, let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appear in the context of word i. Consider the following sample example that showcases how certain aspects of meaning can be extracted directly from co-occurrence probabilities.

Table **1**: This table shows the co-occurrence probabilities for target words ice and $steam$ with selected context words from a 6 billion token corpus. The ratio perfectly captures the word analogy such as "$ice - solid - steam$" and $ice - gas - steam$. It exhibits high/low ratios for related words, and the non-discriminative words such as $water$ and $fashion$ have ratios close to 1 because the co-occurrence probabilities cancel out.

Probability and Ratio	$\boldsymbol{k = solid}$	$\boldsymbol{k = gas}$	$\boldsymbol{k = water}$	$\boldsymbol{k = fashion}$
$\boldsymbol{P(k\|ice)}$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$\boldsymbol{P(k\|steam)}$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\boldsymbol{P(k\|ice)/P(k\|steam)}$	8.9	8.5×10^{-2}	1.36	0.96

Consider two words i and j that exhibit a particular aspect of interest; for concreteness, suppose we are interested in the concept of thermodynamic phase, let us assume $i = ice$ and $j = steam$. For words k related to ice but not steam, say $k = solid$, we expect P_{ik}/P_{jk} will be large. Similarly, for words k related to steam but not ice, say $k = gas$, we expect the ratio to be small. For words k like $water$ or $fashion$, that are either related to both ice and steam, or to neither, the ratio should be close to one. Table-1 shows these probabilities and their ratios for a large corpus, and their numbers confirm the expectations. Compared to the raw probabilities, the ratio is better able to distinguish relevant words ($solid$ and gas) from irrelevant words ($water$ and $fashion$), and it is also better able to discriminate between the two relevant words. Now, the co-occurrence probabilities are encoded by considering three words at a time. For a function F that encodes the relationship between words i, j and k, the word vectors w_i, w_j, w_k should be such that they satisfy equation (12).

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}} \tag{12}$$

$$F(w_i - w_j, w_k) = \frac{P_{ik}}{P_{jk}} \tag{13}$$

$$F\left(\left(w_i - w_j\right)^T w_k\right) = \frac{P_{ik}}{P_{jk}} \tag{14}$$

$$F\left(\left(w_i - w_j\right)^T w_k\right) = \frac{F(w_i^T w_k)}{F(w_j^T w_k)} \tag{15}$$

$$F(w_i^T w_k) = P_{ik} = \frac{X_{ik}}{X_i} \tag{16}$$

$$w_i^T w_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{17}$$

Though the number of possibilities for F is vast, it should follow specific properties. First, it should satisfy equation (13) to encode the information present in the ratio P_{ik}/P_{jk} to the word-vector space. Since vector spaces are inherently linear structures, vector difference is the most natural way to encode this property. To preserve the linear structure when using complex functions such as neural networks, first, a dot-product is introduced to avoid the mismatch between dimensions. Further, the symmetry between word co-occurrence counts can also be maintained with the dot-product, as shown in equation (14), which can be solved by equation (15). The solution to equation (16) is $F = \exp$ or $\log$, as shown in equation (17). Finally, the model is trained with a weighted least squares objective that is summed up over the entire vocabulary.

The complexity of the model is much better than the worst-case $\mathcal{O}(|V|^2)$, and it does somewhat better than the on-line window-based methods like skip-gram and CBOW which scale like $\mathcal{O}(|C|)$ where, $|V|$ and $|C|$ are vocabulary size and corpus size respectively. Pre-trained word

vectors on Wikipedia 2014, and Gigaword 5[2] corpus consisting of 50, 100, 300-dimensional word vectors for 6 billion tokens are released online. It is a common practice to initialize word embeddings used in seq2seq models with these pre-trained ones rather than a random initialization and training from scratch. Since the GloVe [21] word embeddings are trained on a huge corpus, we can assume that our training dataset is sampled from this corpus.

Encoder-Decoder Model

The vanilla RNN can easily map a sequence to another sequence whenever the alignment between inputs and outputs is known ahead of time; for example, Machine Translation for similar languages. However, it is not clear how to apply an RNN to problems whose input and output sequences have different lengths with complicated and non-monotonic relationships.

A simple strategy for general sequence learning is to use an RNN to map the input sequences to a fixed-sized latent vector that essentially captures the input semantics and then to map the latent vector to the target sequence with another RNN [22]. Though RNN is provided with all the relevant input semantics, decoder RNN faces difficulty due to long term dependencies [18]. However, the LSTM [23] is known to learn problems with long-range temporal dependencies; therefore, an LSTM may perform better than a vanilla RNN.

The goal of LSTM-LSTM model is to estimate the conditional probability $p(y_1, y_2, \dots, y_{T_{out}} | x_1, x_2, \dots, x_{T_{in}})$ where, $(x_1, x_2, \dots, x_{T_{in}})$ is an input sequence of length T_{in} and $(y_1, y_2, \dots, y_{T_{out}})$ is its corresponding output sequence of length T_{out}. First, an encoder obtains a

[2] https://catalog.ldc.upenn.edu/LDC2011T07

fixed dimensional representation h of the input sequence $(x_1, x_2, \dots, x_{T_{in}})$ given by the last hidden state of the LSTM. Then a decoder LSTM initialized with hidden state h is used to compute the probability of the output sequence $y_1, y_2, \dots, y_{T_{out}}$ at each time step given the previous targets; this is called Teacher Forcing.

$$p(y_1, y_2, \dots, y_{T_{out}} | x_1, x_2, \dots, x_{T_{in}}) = \prod_{t=1}^{T_{out}} p(y_t | h, y_1, y_2, \dots, y_{t-1})$$

This model is trained by minimizing the negative log probability of the target sequence T given the source sequence S; this is also called as the cross-entropy loss. so the training objective is

$$-\frac{1}{|\mathcal{S}|} \sum_{(T,S) \in \mathcal{S}} \log p(T|S)$$

Where $\mathcal{S}$ is the training set, once the training is complete, the predictions are computed by finding the most probable output by greedy decoding as below or by using beam search. Figure-4 below shows a typical encoder-decoder model used for NMT task.

$$\hat{T} = \arg\max_{T} p(T|S)$$

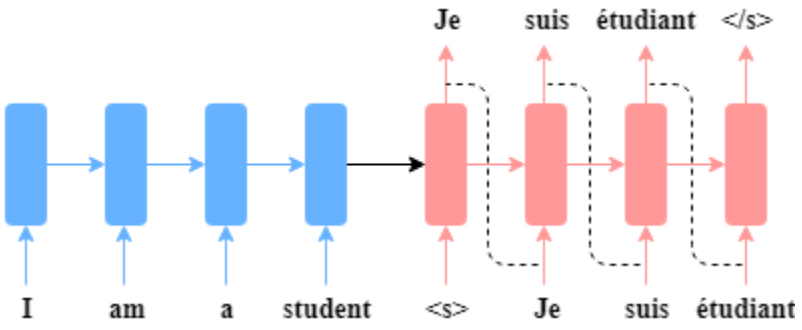

Figure **4**: This is a typical encoder-decoder model used for sequence-to-sequence modeling tasks[3] , for example, a machine translation model from English to French. English words are fed to an encoder RNN at each time step to obtain a semantic representation of the English sentence as the final hidden state. The decoder RNN is then initialized with the final hidden state to use it while

[3] https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention

decoding to French. **<s>** and **</s>** are the start and end token respectively. The dotted lines show greedy decoding where the previous prediction is fed as input in the next time-step, unlike in teacher forcing where the ground truth input is used.

Attention Mechanism

A potential issue with the encoder-decoder model described above is that the neural network needs to be able to compress all the necessary information from the source sequence into a fixed-length vector to help the decoder. This may make it difficult for the neural network to cope with long sequences, especially those that are longer than the sequences in the training corpus. Cho et al. [24] showed that indeed, the performance of a basic encoder-decoder model deteriorates rapidly as the length of an input sentence increases.

To address this issue, Bahdanau et al. [10] introduced an extension to this encoder-decoder model, which learns to align and translate jointly. Each time a word is generated by the decoder, it retrieves the most relevant information from the encoder by softly searching the positions where the information is concentrated. The distinguishing feature of this method is instead of encoding all the source sequence into a single vector, it retrieves a context vector relevant to each decoding time step, and therefore, the conditional probability is modified with this additional dependency as follows:

$$p(y_t|y_1, y_2, \dots, y_{t-1}, x) = g(y_{t-1}, s_t, c_t)$$

Where, c_t is a context vector and s_t is the LSTM hidden state for time t computed as follows:

$$s_t = LSTM(s_{t-1}, y_{t-1}, c_t)$$

The context vector c_t depends on a sequence of annotations $(h_1, h_2, \dots, h_{T_{in}})$ to which an encoder maps the input sequence $(x_1, x_2, \dots, x_{T_{in}})$. Each annotation h_t contains the information about the whole input sequence with a strong focus on the parts surrounding the t-th word of the input sequence. The context vector c_t is then computed as a weighted sum of these annotations h_t:

$$c_t = \sum_{i=1}^{T_{in}} \alpha_{ti}\, h_i$$

The weight of each annotation h_i is computed by

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{T_{in}} \exp(e_{tj})}$$

Where e_{ti} is an alignment score quantifying how well the inputs around position i and output at position t match. The score is based on the LSTM hidden state s_{t-1} just before emitting y_t and the i-th annotation h_i of the input sentence. For the alignment model, Bahdanau et al. [10] use only a single feedforward neural network as follows:

$$e_{ij} = a(s_{i-1}, h_j)$$

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

whereas, Luong et al. [11] proposed a content-based global and local attention that uses either *dot*, a simple dot product between states; *general*, a dot product between linear transform of the encoder state and the decoder hidden state; or *concat*, a dot product between a new parameter v_a and a linear transform of the states concatenated together.

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^T \bar{h}_s & dot \\ h_t^T W_a \bar{h}_s & general \\ v_a^T W_a [h_t, ; \bar{h}_s] & concat \end{cases}$$

Where h_t and $\bar{h}_s$ are current target and source hidden states respectively. In practice, both additive and multiplicative attention results in a similar performance for NMT. Figure-5 depicts the attention mechanism used in a typical NMT task.

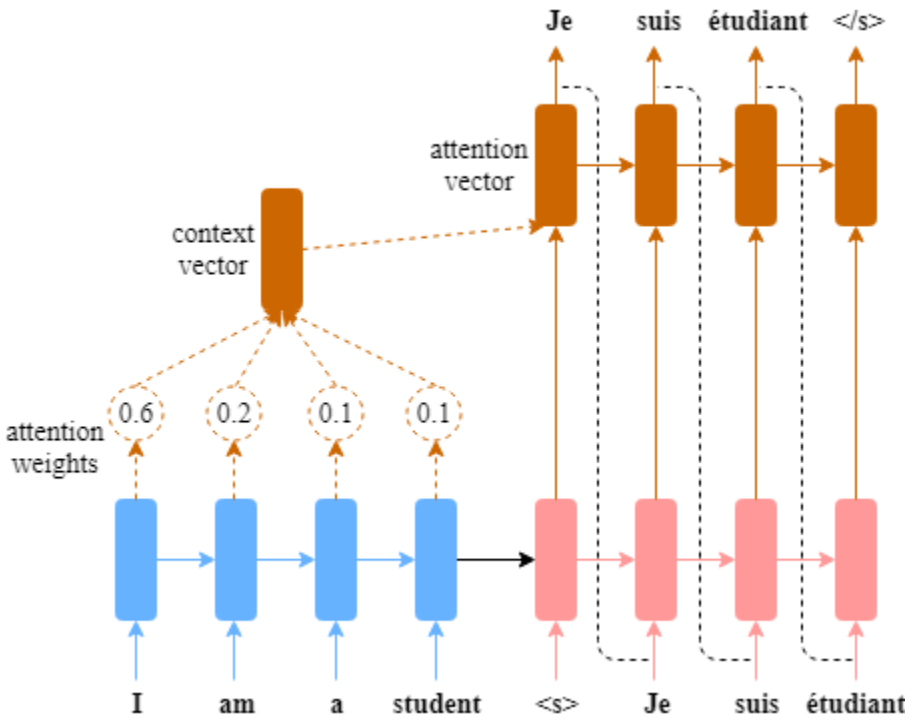

Figure 5: This is the attention mechanism[4]. At each decoder time step, alignment scores are computed using the decoder hidden state and every encoder time step [10], [11]. A weighted sum of the encoder hidden states is computed to obtain a context vector which is then used by the decoder together with its hidden state.

Motivation

Recently, an encoder-decoder approach is used in solving sequential modeling tasks such as machine translation [8], text summarization [1] and question answering [9], etc. The encoder reads the source sequence and maps it to a latent vector. The decoder then uses this hidden information and maps it to the target sequence. An attention mechanism [10] [11] is used to improve its performance for longer sequences [1] [12]. Using the hierarchical attention [13] a single sentence is selected [14] [15] and rewritten to form a concise summary, but this is too sparse as there could be multiple document sentences, usually the highlights of the document, that can be paraphrased

[4] https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention

together as summaries. At the same time, considering all the alignment scores in the hierarchical attention model also failed as its too noisy [3] [4]. We propose a novel memory augmented neural network that is very effective for longer sequences because the source sequence is available to the decoder throughout the decoding process. Our memory is adapted from Neural Semantic Encoders (NSE) [16] after improving its attention mechanism, compose function, and using these changes, we finally propose the Hierarchical NSE.

Chapter 2

Related Work

As discussed earlier, extractive summarization techniques select a few words or sentences from the source document to form concise summaries. A few earlier approaches used sentence-based topic models [25] to represent the summaries, intuitively, it helps to interpret the topics discussed in the text. Usually, topic modeling is done using Term Frequency (TF), Inverse Document Frequency (IDF) statistics, latent semantic analysis or sometimes a Bayesian approach is preferred [26] such as the Latent Dirichlet Allocation (LDA). More recently, Recurrent Neural Networks (RNNs) have been widely used for sequence modeling tasks such as the text summarization. Long Short-Term Memory (LSTM) is a type of RNN that can retain memory for longer sequences and have proven to perform very well in the recent past. Typically, a sequence modeling architecture consists of an encoder LSTM to obtain the semantics of the input, for example, the meaning of a sentence in German in the case of Neural Machine Translation (NMT). The encoded information is used by a decoder to obtain a sequence of words in English that represents the same semantics. This network is trained using the cumulative sum of cross-entropy loss for the entire sequence whose objective is to maximize the likelihood of the corresponding gold summary. Sutskever et al. [8] proposed an encoder-decoder model for NMT that showed state-of-the-art results on short sentences but failed to reach the expectations when used for longer sentences. Though LSTM is designed to memorize an entire sequence, it is observed to be ineffective when it comes to longer sentences. To solve this problem, Bahdanu et al. [10] proposed an attention mechanism that enables the decoder to focus on a small context window. It retrieves a context vector summarizing this context window from the input sequence that a human would typically focus. Inspired by this attention model, various encoder-decoder type architectures were

adapted to the text summarization task. These proposed methods can be classified as supervised learning approaches and reinforcement learning approaches.

Supervised learning approaches

An encoder-decoder, together with an attention mechanism, was first used by Rush et al. [1] for text summarization. Though the encoder-decoder models gave a state-of-the-art performance for NMT, the maximum sequence length used is just 100 tokens. Typical document lengths in text summarization vary from 400 to 800 tokens, and LSTM will not be effective due to the loss in memory over time for very long sequences. To mitigate this effect, Nallapati et al. [4] used hierarchical attention, initially proposed for document classification, for extracting sentences and words. In hierarchical attention, the assumption is that a document is formed sentences and these sentences are formed from words. So, a word LSTM is used to encode (decode) words, and a sentence LSTM is used to encode (decode) sentences. The use of two LSTMs separately for words and sentences improves the model's ability to retain its memory for longer sequences. A hierarchical model consisting of a feature-rich encoder incorporating position, Named Entity Recognition (NER) tag, Term Frequency (TF) and Inverse Document Frequency (IDF) scores are explored by Nallapati et al. [4]. Since an RNN is a sequential model, one time-step needs all of the previous time-steps to have computed before and is slow due to a lack of parallel processing. Chopra et al. [12] increase the speed of the encoder by using convolutional layers coupled with an attention mechanism. Since the input to an RNN is fed sequentially, it is expected to capture the positional information. But both works ([4] and [12]) found positional embeddings to be quite useful for reasons unknown; usually, a one-hot or a sinusoidal embedding is concatenated with the word embeddings to encode the positional information. An extractive summarization model is

proposed by Nallapati et al. [3] which classifies sentences based on content, saliency, novelty, and position.

Due to computational limitations, word vocabularies with typical sizes ranging from 50,000 to 100,000 are used for text summarization. So, many proper nouns, dates, numbers, etc. are usually not found in the vocabulary, and unknown token [UNK] is used to replace these out-of-vocabulary (OOV) words. Since the model is trained on many [UNK] tokens, there is a greater likelihood of the decoder predicting [UNK] tokens. Therefore, it is challenging to interpret the model predictions when [UNK] tokens are present. The decoder often misplaces similar words, for example, *London* in place of *New York*, *2000* in place of *1994* because of the proximity of the word vectors in the semantic space and also the inability of the decoder to copy tokens from the input sequence. For this reason, though the encoder-decoder model generates good abstractive summaries, it fails to copy input words when necessary, especially proper nouns and numbers. To overcome the issue of copying [1], Vinyals et al. [27] proposed pointer-networks to replace the [UNK] tokens with the most attentive word from the encoder.

See et al. [2] further modified the pointer-mechanism by adding the functionality of deciding whether to generate a word from the vocabulary or copy the most attentive word from the input. Attention models for longer sequences tend to be repetitive due to the decoder repeatedly attending to the same position from the encoder. To mitigate this issue, See et al. [2] proposed coverage mechanism; a causal framework in which present attention distribution is dependent on all the previous attention distributions. Additionally, coverage loss; a cumulative sum of attention distributions is added to the cross-entropy training objective; the negative log-likelihood of gold summary sequence to help the model in learning coverage parameters. However, the pointer generator and the coverage model [2] are still highly extractive; copying the whole article sentences

35% of the time. Paulus et al. [28] introduced an intra-attention model in which attention also depends on the predictions from previous time-steps.

Reinforcement learning approaches

One of the main issues with sequence-to-sequence models is that optimization using cross-entropy objective does not always provide excellent results because the models suffer from a mismatch between training objective; the cross-entropy loss and evaluation procedure; different metrics such as ROUGE [29], METEOR [30], etc. are used to quantify the similarity between predicted and gold summaries. A popular algorithm to train the decoder is the teacher-forcing [31] algorithm that minimizes the negative log-likelihood (cross-entropy loss) at each decoding time step given the previous ground-truth outputs. But during the testing stage, the prediction from the previous time-step is fed as input to the decoder instead of the ground truth. This exposure bias [32] results in error accumulation over each time step because the model has never been exposed to its predictions during training. Instead, recent works show that summarization models can be trained in a reinforcement learning (RL) setting with ROUGE [29] score as the reward.

In RL [33], a sequential Markov Decision Process (MDP) is considered in which agents interact with an environment ε over discrete time steps t. Let $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, s_0, \gamma, T)$ represent this discrete time-horizon discounted MDP where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition probability distribution, $\mathcal{R}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\mathcal{P}: s_0 \to \mathbb{R}_+$ is the initial state distribution, $\gamma \in [0, 1]$ a discount factor, and T is the horizon. In our case, the goal of the agent is to excel at generating text that receives a higher ROUGE score. The idea is that given the environment state at time t as s_t, the agent picks an action $\hat{y}_t \in \mathcal{A}$, according to a stochastic policy $\pi: \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ and receives a reward r_t for that action. The cumulative

discounted sum of rewards $R_t = \mathbb{E}_\pi[\sum_{\tau=0}^{T} \gamma^\tau r_\tau]$ is the objective function optimized by the policy π where the discounting factor γ controls the trade-off between immediate rewards (a single sentence ROUGE) and future rewards (total summary ROUGE). Under the policy π, we can define the values of state-action pair $Q(s_t, y_t)$, the value $V(s_t)$ and calculate the loss as follows:

$$Q_\pi(s_t, y_t) = \mathbb{E}[r_t | s = s_t, y = y_t]$$

$$V_\pi(s_t) = \mathbb{E}_{y \sim \pi(s)}[Q_\pi(s_t, y_t)]$$

$$\mathcal{L}_\theta = -\mathbb{E}_{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T \sim \pi_\theta}[r(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)]$$

Intuitively, value function V measures how good the model could be when it is in a specific state. However, the Q function measures the value of choosing a specific action when we are in a given state. $\hat{y}_t$ is the action chosen by the model at time t and $r(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ is the reward associated with actions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T$. In practice, the loss function is usually approximated with just one sample from the policy distribution. Hence the derivate of the above loss function is as follows:

$$\nabla_\theta \mathcal{L}_\theta = -\mathbb{E}_{\hat{y}_{1\ldots T} \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(\hat{y}_{1\ldots T})\, r(\hat{y}_{1\ldots T})]$$

We can now use chain rule and backpropagate the gradient to the policy inside the expectation; hence, they are called policy-gradient algorithms. If the expectation of this gradient estimator is zero, then it is unbiased.

Henß et al. [7] made such an earlier attempt by using Q-learning for single-and multi-document summarization. But their model cannot approximate state values precisely. Ling and Rush [15] proposed a coarse-to-fine hierarchical attention model to select a salient sentence using sentence attention and feed it to the decoder. Since hard-attention; selecting the maximally attended sentence is non-differentiable, REINFORCE [34]; an unbiased policy gradient algorithm, is used to train the sentence selection module. Narayan et al. [35] used REINFORCE [34] to rank sentences for extractive summarization. Celikyilmaz et al. [6] proposed deep communicating agents that

operate over small chunks of a document, which is learned using a self-critical [36] training approach consisting of intermediate rewards. Chen and Bansal [14] used advantage actor-critic (A2C); a synchronous variant of A3C [37] method to extract sentences followed by a decoder to form abstractive summaries. Our model does not make their limiting assumption that a summary sentence is an abstracted version of a single source sentence. Paulus et al. [28] trained their intra-attention model using a self-critical policy gradient algorithm [36]. Though an RL objective gives high ROUGE [29] scores, the output summaries are not readable by humans. To mitigate this problem, Paulus et al. [28] used a weighted sum of supervised learning loss and RL loss.

Humans first form an abstractive representation of sentences in mind (meaning) and then try to put it into words while communicating. Though it seems intuitive that there is a hierarchy from sentence representation to words, as observed by both Nallapati et al. [4] and Ling and Rush [15], these hierarchical attention models failed to outperform the simple attention model [1]. Unlike feedforward networks, RNNs are expected to capture the input sequence order. But strangely positional embeddings are found to be effective [4], [12], [15]and [3]. This work demonstrates a few approaches that are explored to solve these issues and improve the performance of neural models for abstractive summarization.

Chapter 3

Proposed Methods

A Natural Language Processing (NLP) task with sequential data such as the Neural Machine Translation, Sentiment Analysis, Text Summarization, etc. are typically modeled using an RNN such as an LSTM or the GRU (Gated Recurrent Unit). Most of the simple and also advanced encoder-decoder models were primarily proposed for Machine Translation where typical sequence length is not more than 100 tokens. The LSTM does not have to capture a global context, but instead, it would suffice to translate on a word-by-word basis or in the worst-case over a small context window. Unlike in machine translation, for text summarization, the LSTM has to capture the semantics of each word, document sentence and also the whole document on some level and later condense these semantics into a summary. To overcome these difficulties, several studies have explored ways of extending neural networks with an external memory [38], [39], [40]. Neural Semantic Encoders (NSE) [16] is one such memory augmented neural network that we have explored and adapted for text summarization. NSE offers several desirable properties such as the variable-sized encoding memory and also word compositionality that can inherit both temporal and hierarchical nature of human language.

Problem Formulation

Let $D = \{d_1, d_2, \dots, d_N\}$ be the set of document sentences where each sentence d_i is a set of words and $S = \{s_1, s_2, \dots, s_M\}$ be the set of summary sentences. In general, most of the sentences in D are a continuation of another sentence or relate to each other in terms of factual details or pronouns used, etc. So, dividing the document into multiple paragraphs as in Celikyilmaz et al. [6] leaves out

the possibility of a sentence-level dependency between say start and end of a document. Similarly, abstracting a single document sentence as followed in Chen and Bansal [14] cannot include related information from multiple document sentences. A more plausible assumption seems to be that each summary sentence is a compressed version of a few document sentences. Mathematically,

$$\forall s \in S, \exists d_1, d_2, \ldots, d_K \in D \; s.t \; C(d_1, d_2, \ldots, d_k) = s$$

Where C is a compressor that we intend to learn by minimizing the cross-entropy loss between predicted and gold summaries. Figure-6 depicts the fundamental idea when using a sequence-to-sequence architecture. For a sentence s in summary, the representations of all the related document sentences $d_1, d_2, \ldots, d_K$ are expected to form a cluster which represents a highlight of the document. As you will discover later, this highlight is computed using a weighted combination of all the document sentence representations in which the document representations nearer (farther) to the relevant cluster are assigned higher (lower) weights by the model. In sparse scenarios, a single (few) sentence(s) might be assigned a very high weight.

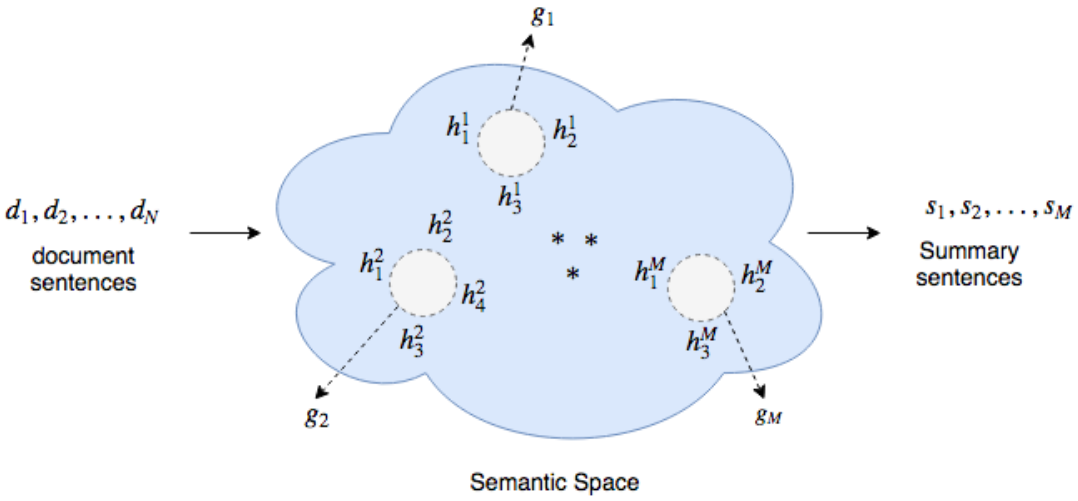

Figure **6**: Summarization model: assume that all semantics of document sentences belong to different clusters $\{[h_1^1, h_2^1, h_3^1], [h_1^2, h_2^2, h_3^2], \ldots, [h_1^M, h_2^M, h_3^M]\}$. Each individual cluster of $G = \{g_1, g_2, \ldots, g_M\}$ now represent the summary semantics of all the document sentences belonging to that cluster. A decoder can then be used to obtain words from these cluster semantics.

Neural Semantic Encoder

Neural Semantic Encoder (NSE) [16] is a memory augmented neural network equipped with a variable-sized encoding memory that allows the model to access the entire input sequence at each time-step. In an LSTM, the short-term memory weights are a part of the training parameters, and therefore, it imposes practical difficulties in training and modeling long sequences. Unlike an LSTM, the availability of input sequences via memory helps in efficiently modeling long-term temporal dependencies over time. The encoding memory that maintains the input sequence is evolved through time using *read*, *compose*, and *write* operations. NSE sequentially processes the input and exhibits word compositionality inheriting both temporal and hierarchical nature of human language. NSE can read and write to any number of relevant encoding memories simultaneously, and can also support knowledge and representation sharing by providing access to a shared encoding memory. NSE is robust, flexible, and suitable for practical Natural Language Understanding tasks and can be trained easily by any gradient descent optimizer.

The NSE memory slots are first initialized with the input sequence. Then, at each time-step, NSE performs three primary operations. First the input embedding vector x_t at the current time-step is passed through an LSTM and retrieves a memory slot $m_{r,t}$ that is semantically aligned with the present input word w_t. The attention mechanism is used to read a memory slot location r ($1 < r < l$), defined by the attention distribution z_t. The compose function then composes new semantics by combining the retrieved memory slot $m_{r,t}$ with the current input x_t. The write function then projects the composed vector onto the encoding memory space and writes the new information into the accessed memory slot. A seq2seq schema using an NSE is shown in Figure-7.

Let x_t be the raw word embedding vector at the current time-step, $f_r^{LSTM}, f_c^{MLP}, f_w^{LSTM}$ be the read, compose, and write functions, respectively. Following equations describe the functionality of NSE:

$$o_t = f_r^{LSTM}(x_t) \tag{18}$$

$$z_t = softmax(o_t^T M_{t-1}) \tag{19}$$

$$m_{r,t} = z_t^T M_{t-1} \tag{20}$$

$$c_t = f_c^{MLP}(o_t, m_{r,t}) \tag{21}$$

$$h_t = f_w^{LSTM}(c_t) \tag{22}$$

$$M_t = M_{t-1}(\mathbf{1} - (z_t \otimes e_k)^T) + (h_t \otimes e_l)(z_t \otimes e_k)^T \tag{23}$$

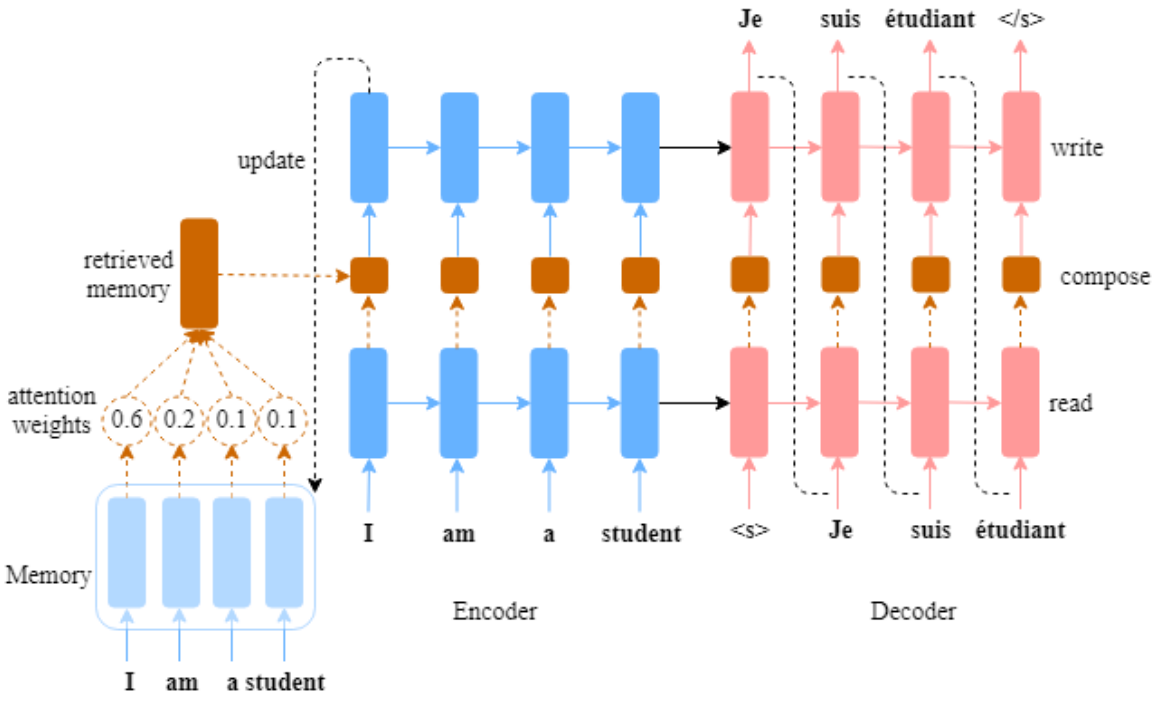

Figure **7**: Neural Semantic Encoder: Memory is initialized with word embeddings. The input is read at each time-step, followed by retrieval of semantically associated information from memory, the composition of new semantics and finally, update the memory.

Where, $e_l \in R^l$ and $e_k \in R^k$ are vectors of ones, $\mathbf{1}$ is a matrix of ones, $\otimes$ denotes the outer product which duplicates its left vector l or k times to form a matrix. The read function f_r^{LSTM} in equation (18) sequentially maps the word embeddings to the internal space of memory M_{t-1} to obtain the hidden states o_t. Now, the NSE uses a simple dot-product soft-attention mechanism with o_t as the key to retrieve content from semantically aligned slots of the memory. z_t in equation (19) is the attention distribution and $m_{r,t}$ in equation (20) is the retrieved memory vector. Equation (21) concatenates the current hidden state o_t with the retrieved memory $m_{r,t}$ to pass through a Multi-Layer Perceptron (MLP) and obtain the composed vector c_t. This composed vector is mapped to the encoder output space using an LSTM as in equation (22). Finally, the newly formed representation h_t is written to the memory locations as in equation (23), first, the slot information used for retrieval is erased as per z_t and new information h_t is written. NSE performs this iterative process until all words in the input sequence are read.

The encoding memories $\{M\}_{t=1}^{T}$ and output states $\{h\}_{t=1}^{T}$ are used for tasks such as text summarization by feeding the output states through an output projection layer. The key here for text summarization is that NSE has any-time access to the entire fixed-length input sequence stored in the encoding memory; this overcomes the shortcomings of using plain LSTMs for longer sequences. Although NSE performed well for machine translation, just a dot-product attention mechanism is too simplistic for text summarization. Also, compose function can be improved using an LSTM to keep track of previously composed vectors and guide novel composition.

Hierarchical NSE

When humans read a document, we organize it in terms of word semantics, sentence semantics, and then document semantics. In a text summarization task, after reading a document,

sentences that have similar meanings or continual information are grouped together and then expressed in words. Such a hierarchical model was first introduced by Yang et al. for document classification [13] and later explored for text summarization [4] [3]. In this work, we propose to use a hierarchical model with improved NSE to exploit both augmented memory and also the hierarchical document representation. We use a separate memory for each sentence to represent all the words of a sentence and a document memory to represent all sentences. Word memory composes novel words, and document memory composes novel sentences in the encoding process that can be later used while decoding to summaries.

Let $D = \left\{w_{j=1}^{T_{in}}\right\}_{i=1}^{S_{in}}$ be the input document sequence, where S_{in} is the number of sentences in a document and T_{in} is the number of words per sentence. Let $\{M_i^s\}_{i=1}^{S_{in}}, M_i^s \in \mathcal{R}^{T_{in} \times D}$ be the encoding memories of each sentence in the input sequence and $\{M^d\}, M^d \in \mathcal{R}^{S_{in} \times D}$ be the document encoding memory. At each time-step, an input token x_t is read and it is used to retrieve aligned content from both corresponding sentence M_{t-1}^s and document memories M_{t-1}^d. Please note that the retrieved document memory, which is a weighted combination of all the sentence representations forms a highlight. After composition, both the sentence and document memories are written simultaneously. This way, the words are encoded with contextual meaning, and also new simpler sentences are formed. The functionality of the model is as follows:

$$o_t = f_r^{LSTM}(x_t) \tag{24}$$

$$z_t^s = softmax\left(v_a^T \tanh(W_a M_{t-1}^s + U_a o_t + b_{attn})\right) \tag{25}$$

$$z_t^d = softmax\left(v_a^T \tanh\left(W_a M_{t-1}^d + U_a o_t + b_{attn}\right)\right) \tag{26}$$

$$m_{r,t}^s = z_t^s M_{t-1}^s \tag{27}$$

$$m_{r,t}^{d} = z_t^d M_{t-1}^{d} \tag{28}$$

$$c_t = f_c^{LSTM}\left(o_t, m_{r,t}^{s}, m_{r,t}^{d}\right) \tag{29}$$

$$h_t = f_w^{LSTM}(c_t) \tag{30}$$

$$M_t^s = M_{t-1}^s(\mathbf{1} - (z_t^s \otimes e_k)^T) + (h_t \otimes e_l)(z_t^s \otimes e_k)^T \tag{31}$$

$$M_t^d = M_{t-1}^d\left(\mathbf{1} - \left(z_t^d \otimes e_k\right)^T\right) + (h_t \otimes e_l)\left(z_t^d \otimes e_k\right)^T \tag{32}$$

Where, v_a, W_a, U_a, b_{attn} are learnable parameters. Equation (24) is the normal read operation to obtain the hidden states of the LSTM. Equation (25), (26) are the additive attention mechanisms that are introduced by Bahdanu et al. [10] to obtain the sentence and document attention distributions. Useful information in the context of the current word is retrieved from the sentence and document memories as per equations (27) and (28). The retrieved vectors are later concatenated along with the read hidden state to pass through an LSTM as per equation (29); this is different from the standard NSE where a simple MLP is used. We find this useful because the composition at the current time step will depend on what the model has composed before to maintain novelty. Finally, the write vector computed as per equation (30) is projected back to the memory subspace and updated as per equations (31) and (32). During the decoding phase, the word memories are concatenated to get a single word memory $\{M^s\}, M^s \in \mathcal{R}^{(T_{in} * S_{in}) \times D}$ and used in the same fashion as a single NSE.

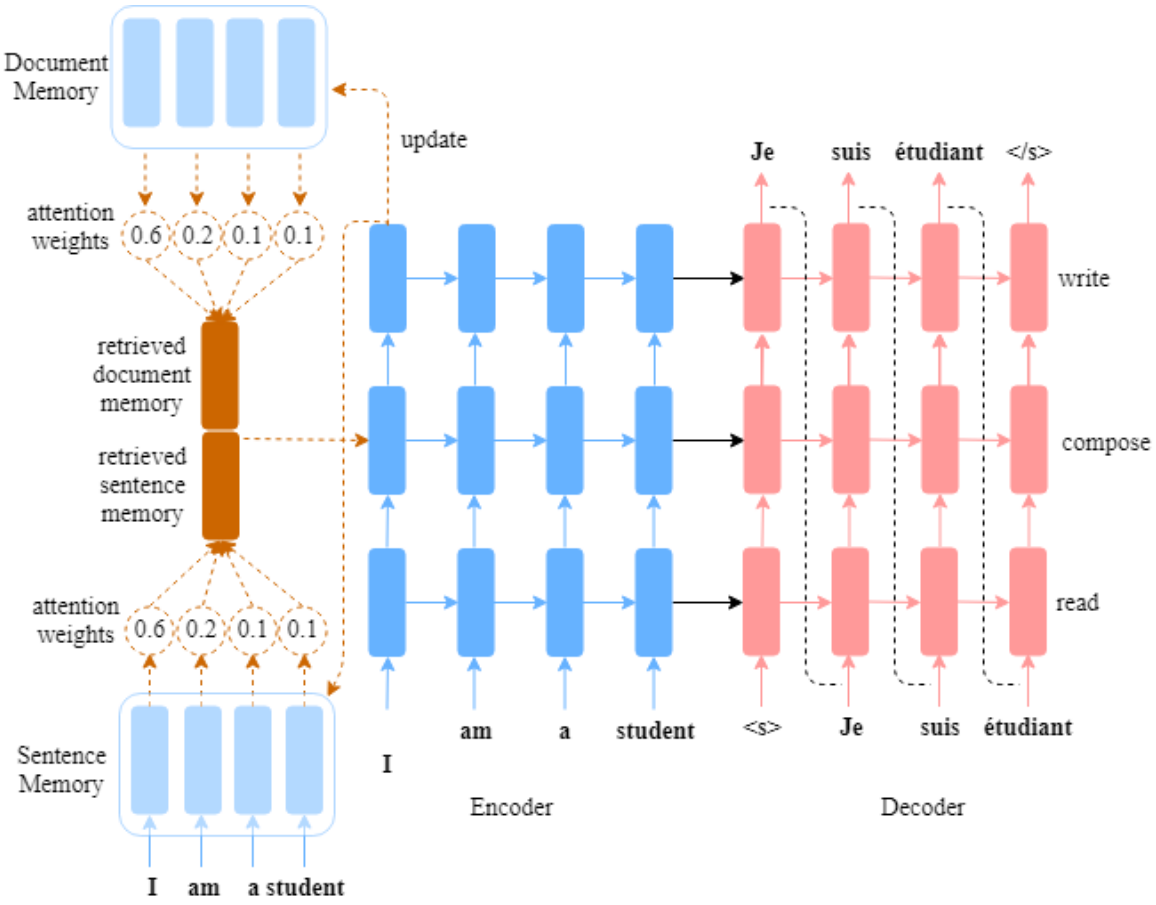

Figure **8**: Hierarchical Neural Semantic Encoder: S sentence memories each of size T are maintained for whole document representation. Sentence memories are initialized with corresponding input sequences, and document memory is initialized with sentence representations, which is the average of all word vectors in a sentence. At each time-step when input is read, the semantically associated information is retrieved from both document (a highlight), and sentence memories, followed by composition and finally, memories are updated.

Multi-Head Attention

Instead of performing a single attention function with D dimensional queries and values, Vaswani et al. [41] proposed a multi-head attention mechanism originally for machine translation. First, the memory and the read vector are linearly projected h times with different learned linear projections to d_h dimensions where, $d_h = D/h$. On each of the h query-value pairs, attention

mechanism is applied in parallel to retrieve h, d_h dimensional vectors. These are concatenated and projected back into the original encoding space.

$$\text{MultiHead}(M_t, r_t) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where, head}_i = \text{Attention}(M_t W_i^m, r_t W_i^r)$$

Where the projections are parameter matrices $W_i^m \in \mathcal{R}^{D \times d_h}, W_i^r \in \mathcal{R}^{D \times d_h}$ and $W^O \in \mathcal{R}^{D \times d_h}$. Since the memory is updated in each time-step, it falls into a feedback loop if it is projected in every time-step as the target keeps on shifting. So, memory is projected only once into the h, d_h dimensional subspaces and is also updated there itself (without projecting back). We have used 512-dimensional word embeddings with 8 heads for the Hierarchical NSE. Please note that due to the reduced dimension of each head, the total computation cost is similar to that of single-head attention with full dimensionality.

Pointer Generator Mechanism

Due to the limited size of word vocabulary, out-of-vocabulary (OOV) words are replaced with [UNK] tokens. *pointer-networks* [27] facilitate the ability to copy words from the input sequence to the output via *pointing*. Later, See et al. [2] proposed a hybrid pointer-generator mechanism to improve upon pointing by retaining the ability to generate new words. It points to the words from the input sequence and generates new words from the vocabulary. The attention distribution z_t^s, retrieved sentence memory $m_{r,t}^s$, retrieved document memory $m_{r,t}^d$ and write vector h_t are calculated as in the hierarchical NSE. A generation probability $p_{gen} \in [0, 1]$ is calculated using the retrieved memories, attention distribution, current input hidden state o_t and write state h_t as follows:

$$p_{gen} = \sigma\left(W_{m,s}^T m_{r,t}^s + W_{m,d}^T m_{r,t}^d + W_h^T h_t + W_o^T o_t + b_{ptr}\right)$$

Where, $W_{m,s}, W_{m,d}, W_h, W_o, b_{ptr}$ are learnable parameters, and σ is the sigmoid function. Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from the attention distribution z_t^s. For each document, we maintain an auxiliary vocabulary of OOV words in the input sequence. We obtain the following final probability distribution over the total extended vocabulary:

$$p(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w=w_i} z_i^{s,t}$$

Note that if w is an OOV word, then $P_{vocab}(w)$ is zero; similarly, if w does not appear in the source document, then $\sum_{i:w=w_i} z_i^{s,t}$ is zero. The ability to produce OOV words is one of the primary advantages of pointer-generator mechanism. We can also use a smaller vocabulary size and thereby speed up the output projection and softmax layer computation. Figure-9 shows the pointer-generator mechanism used for NMT task.

Coverage

Repetition is a common problem for sequence-to-sequence models [42] [43] [44], and the problem gets worsened when generating multi-sentence text. To tackle this problem, Tu et al. proposed a coverage model for machine translation task. See et al. [2] later adapted this coverage to text summarization task to penalize repetitions by keeping track of previous attention distributions. A coverage vector c^t, which is the sum of attention distributions until the current time step, is maintained.

$$c^t = \sum_{t'=1}^{t-1} z_{t'}$$

c^t is an unnormalized distribution over all the words in the source document that represents the degree of coverage received from the attention mechanism so far. c^0 here is initialized with zeros, because none of the source document is covered in the first time-step.

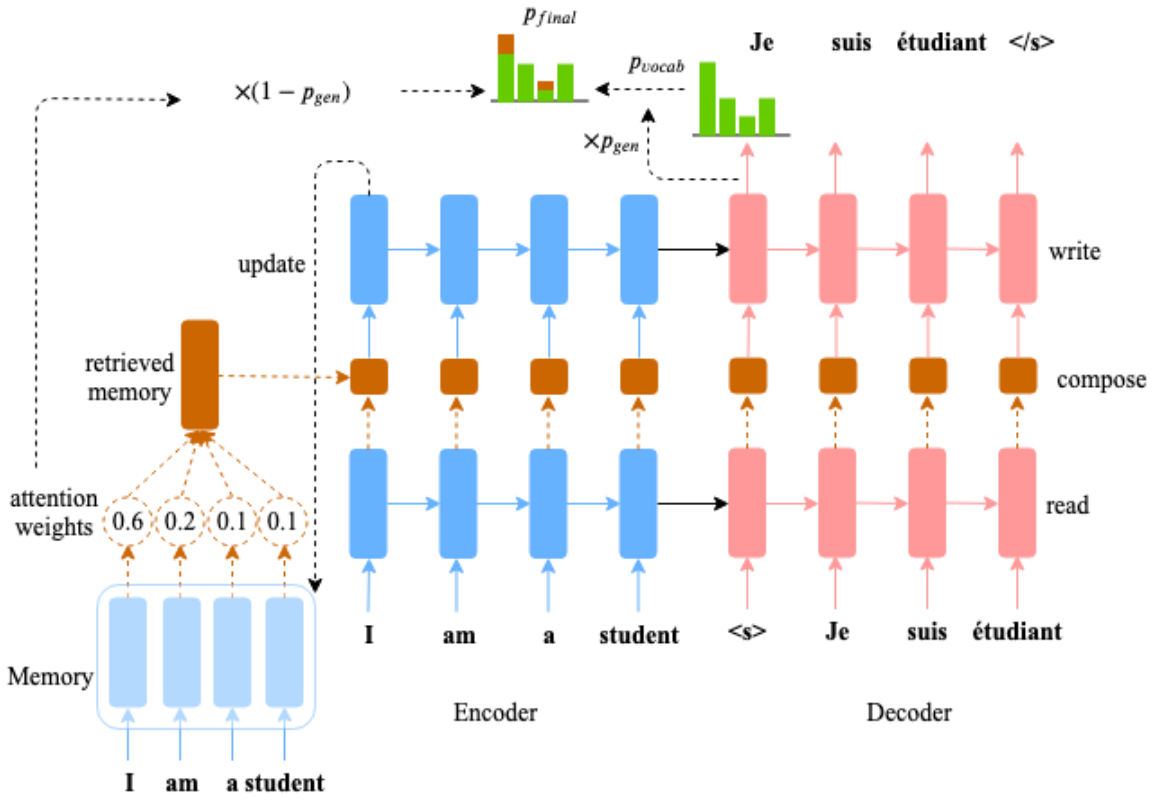

Figure **9**: Pointer Generator Mechanism: OOV words in the input sequence together form an extended vocabulary. At each decoder time-step, additional probability p_{gen} is calculated along with p_{vocab} that facilitates copying and also retains the ability to generate a new word. p_{final} is then computed using Bayes theorem.

The attention mechanism should also adapt as per the coverage. So, we add an extra dependency to the score calculation as follows:

$$z_t^s = v_a^T \tanh(W_a M_{t-1}^s + U_a o_t + w_c c^t + b_{attn})$$

Where, w_c is a learnable parameter vector. This ensures that the attention mechanism is informed by a reminder of its previous decisions expressed as a summation in the coverage vector c^t. Intuitively we can see if the parameter w_c learns negative values, as the coverage vector c^t gets

accumulated the attention scores of the respective slots will be reduced. It is also necessary to provide the objective to coverage model in the form of coverage loss added to the supervised loss.

$$\text{cov\_loss}_t = \sum_i \min\left(z_i^t, c_i^t\right)$$

Here, note that the coverage loss is bounded; in particular $\text{cov\_loss}_t \leq \sum_i z_i^t = 1$. Finally, the supervised and coverages losses are weighted by λ to form the total loss as follows:

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_{i=1}^{T_{in}} \min\left(z_i^t, c_i^t\right)$$

Chapter 4

Evaluation

This chapter will go through the evaluation metrics, datasets used, and the results obtained from all the experiments and a comparison to the existing methods.

ROUGE

ROUGE stands for Recall-Oriented study for Gisting Evaluation [29]. Traditionally evaluation of a summary involves human judgments of different quality metrics such as coherence, conciseness, grammaticality, readability, and content. But it is a time-consuming process and requires a lot of human labor, for example, a simple manual evaluation of summaries on a large scale over a few linguistic quality questions and content coverage as in the Document Understanding Conference (DUC) [45] would require over 3000 hours of human effort. Therefore, an automatic quality evaluation metric for various NLP tasks is beneficial. BLEU [46] is one such metric that is found increasingly useful over the years for machine translation evaluation. ROUGE is a similar metric that has been used to evaluate different summarization models. It includes various metrics such as ROUGE-N and ROUGE-L that quantify the similarity between two summaries most commonly, the predicted and ground truth summary.

ROUGE-N

ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries.

$$ROUGE\_N_{recall} = \frac{\sum_{i=1}^{S} \left| ref_n^{(i)} \cap eval_n \right|}{\sum_{i=1}^{S} \left| ref_n^{(i)} \right|}$$

$$ROUGE\_N_{precision} = \frac{\sum_{i=1}^{S} \left| ref_n^{(i)} \cap eval_n \right|}{\sum_{i=1}^{S} \left| eval_n^{(i)} \right|}$$

Where, $ref_n^{(i)}$ is the set of all n-grams present in the i^{th} reference summary, $eval_n$ is the set of all n-grams present in the candidate summary that is to be evaluated and S is the total number of reference summaries. Though initially, just the recall measure is published, it is a common practice in the text summarization literature to report precision, recall and F1 scores (F1 score is the harmonic mean of precision and recall). Note that the number of reference summaries in the denominator increases as we add more references. This is particularly useful when there are multiple summaries, which is quite intuitive as no two human summaries will be same. So, every time we add more reference summaries, we expand the space of alternative summaries. We can also incentivize the model to output our desired types of summaries by adding those types of reference summaries to the total reference pool. Therefore, the candidate summary that matches with many reference summaries is scored higher. This is similar to the evaluation of an essay writing competition by multiple judges. Table-2 shows an example of calculating ROUGE-2 precision and recall measures.

Table 2: This example demonstrates a calculation of ROUGE-2 score for a given candidate and reference summary. There are four overlapping bigrams (bigrams in green are overlapping, and the ones in red are non-overlapping), therefore, $ROUGE\_2_{recall} = \frac{4}{5}$ and $ROUGE\_2_{precision} = \frac{4}{6}$.

	Candidate	***Reference***
Summary	the cat was found under the bed	the cat was under the bed
Bigrams	the cat	the cat
	cat was	cat was
	was found	was under
	found under	under the
	under the	the bed
	the bed	

ROUGE-L: Longest Common Subsequence

A sequence $Z = [z_1, z_2, \dots, z_n]$ is a subsequence of another sequence $X = [x_1, x_2, \dots, x_m]$, if there exists a strictly increasing sequence $[i_1, i_2, \dots, i_k]$ of indices of X such that for all $j = 1,2, \dots, k$, we have $x_{ij} = z_j$. Given two sequences, X and Y, the Longest Common Subsequence (LCS) of X and Y is a common subsequence with maximum length. LCS was used in the past to measure cognateness between two strings and similarity between two texts in the automatic evaluation of summarization models. For summarization, only the sentence-level LCS is used for evaluation in which a sentence is viewed as a sequence of words. The intuition is that longer the LCS of two summaries, the more similar the two summaries are. Let X be the reference summary of length m and Y be the candidate summary of length n. Then ROUGE-L recall, precision, and F1 score are calculated as follows:

$$ROUGE\_L_{recall} = \frac{LCS(X,Y)}{m}$$

$$ROUGE\_L_{precision} = \frac{LCS(X,Y)}{n}$$

$$ROUGE\_L_F = \frac{(1+\beta^2)ROUGE\_L_{recall}ROUGE\_L_{precision}}{ROUGE\_L_{recall} + \beta^2 ROUGE\_L_{precision}}$$

Where, $\beta = ROUGE\_L_{precision}/ROUGE\_L_{recall}$ and $LCS(X,Y)$ is the length of the longest common subsequence of X and Y. The LCS based F measure is called as the ROUGE-L. Notice that ROUGE-L is 1 when $X = Y$ as expected and 0 when $LCS(X,Y) = 0$. One advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence-level word order as n-grams. ROUGE-L, as defined above, has the property that its value is less than or equal to the minimum of unigram F-measure between X and Y. Unigram recall reflects the proportion of words in X that are also Y; while unigram precision reflects the proportion of words Y that are also in X. Unigram recall and precision count all co-occurring words regardless of their order; while ROUGE-L counts only in-sequence co-occurrences. Table-3 shows an example of the difference between ROUGE-N and ROUGE-L. Using S_1 as the reference and S_2 and S_3 as the candidate summaries sentences, S_2 and S_3 have the same ROUGE-2 score, since they both have one bigram "gunman" that is common with the reference bigrams. In the case of ROUGE-L, S_2 has a score of $3/4 = 0.75$ and S_3 has a score of $2/4 = 0.5$, with $\beta = 1$. Therefore S_2 is a better summary compared to S_3 according to ROUGE-L. However, LCS suffers one disadvantage that it counts only main in-sequence words; therefore, other alternative LCS's and shorter sequences are not reflected in the final score. For example, for the candidate sequence S_4, using S_1 as its reference, LCS counts either "the gunman" or "police killed", but not both; therefore, S_4 has the same ROUGE-L score as S_3. ROUGE-2 would prefer S_4 than S_3. While ROUGE automatic evaluation is simple and easy to calculate, it has its drawbacks. It compares summaries word to word on an n-grams or LCS basis, and it does not take the semantics of the sentences or words into consideration.

An excellent way to solve this is to calculate the quality score based on word-vector similarity rather than word similarity. If a candidate summary is less verbose than the reference summary but covers the same topics as the latter, ROUGE has no way of taking that into account.

Table 3: This table demonstrates the difference between ROUGE-L and ROUGE-N. Using S_1 as the reference, ROUGE-2(S_1, S_2) = ROUGE-2(S_1, S_3) but, ROUGE-L(S_1, S_2) = 3/4, ROUGE-L(S_1, S_3) = 1/2. Whereas, ROUGE-L(S_1, S_3) = ROUGE-L(S_1, S_4), ROUGE-2(S_1, S_3) = 1/3 and ROUGE-L(S_1, S_4) = 2/3. In conclusion, ROUGE-L prefers S_2 compared to S_3 and ROUGE-2 prefers S_4 compared to S_3.

	SENTENCES	**BIGRAMS**	**LCS**
S_1	police killed the gunman	police killed, killed the, the gunman	
S_2	police kill the gunman	police kill, kill the, the gunman	{police, the, gunman}
S_3	the gunman kill police	the gunman, gunman kill, kill police	{the, gunman}
S_4	the gunman police killed	the gunman, gunman police, police killed	{the, gunman}

Dataset

We used the CNN/Daily Mail dataset [4], which is the standard to compare text summarization models. It is produced by modifying an existing corpus that has been used for the task of passage-based question answering. The authors used human-generated abstractive summary bullets from news-stories in *CNN* and *Daily Mail* websites as questions (with one of the entities hidden), and stories as the corresponding passages from which the system is expected to answer the fill-in-the-blank question. The authors crawled the websites to extract and generate pairs of passage and questions from these websites. The summary bullets of each story in the original order are restored to obtain a multi-sentence summary; here each bullet is treated as a sentence. This corpus has 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs, as defined by their scripts. The source document in the training set has 766 words spanning 29.74 sentences on an average while the summaries consist of 53 words and 3.72 sentences [4]. The unique characteristics of this dataset such as long documents, and ordered multi-sentence summaries present interesting challenges, especially because the proven sequence-to-sequence LSTM based models find it hard to learn long-term dependencies in long documents. We have used the same train/validation/test split and examples for a fair comparison with the existing models.

Experiments

For all the plain NSE models, we have truncated the article to a maximum of 400 tokens and the summary to 100 tokens. For the hierarchical NSE models, articles have 20 sentences with 20 words each and summaries are limited to 100 tokens. Shorter sequences are padded with [PAD] tokens to reach the maximum limit, and longer sequences are truncated. For all the models, including pointer-generator model, we use a vocabulary size of 50,000 words for both source and

target. Though some previous works [4] have used large vocabulary sizes of 150,000, since our models have a copy mechanism, smaller vocabulary is enough to obtain good performance. Please note that large vocabularies mean an increase in final computation time. Since memory plays a prominent role in retrieval and update, it is vital to start with a good initialization. We have used 300-dimensional pre-trained GloVe word-vectors to represent the input sequence to a model. Sentence memories are initialized with GloVe word-vectors of all the words in that sentence. Document memories are initialized with vector representations of all the sentences where a sentence is represented with the average of the GloVe word-vectors of all its words. All the models were trained using Adam optimizer with the default learning rate of 0.001. We have not applied any regularization as the usage of dropout, and L2 penalty resulted in similar performance with training time drastically increased.

All the models are trained on 4-NVIDIA Tesla-P100 GPUs with a batch size of 128 (32 examples per GPU). The Hierarchical models process one sentence at a time, and hence attention distributions need less memory, and therefore, a larger batch size can be used, which in turn speeds up the training process. All our models reach optimal cross-entropy loss in just 8 epochs, unlike 33-35 epochs for both Nallapati et al. [4] and See et al. [2]. Once the cross-entropy loss is minimized, coverage loss is enabled to train the coverage parameters from the previous checkpoint. Coverage parameters are not trained together in the initial stage because the coverage loss interferes with the cross-entropy objective, and it deteriorates the performance. I have used TensorFlow, an open-source deep learning framework for implementation and experimentation.

Results

All the models are evaluated using the standard metric ROUGE; we report the F_1 scores for ROUGE-1, ROUGE-2, and ROUGE-L which quantitively represent word-overlap, bigram-overlap, and longest common subsequence between reference summary and the summary that is to be evaluated. We obtain our results using *pyrouge* package[5]. The progress of the project is summarized in Table-4. A direct implementation of NSE performed very poorly due to the simple dot-product attention mechanism. In NMT, a transformation from word-vectors in one language to another one (say English to French) using a mere matrix multiplication is enough because of the one-to-one correspondence between words and the underlying linear structure imposed in learning the word vectors [21]. But it is not the case in text summarization because a word (sentence) could be a condensation of a group of words (sentences). Therefore, using a complex attention mechanism proposed by Bahdanu et al. [10] improved the performance. Both dot-product and additive [10] mechanisms perform similarly for NMT task, but the difference is more pronounced for text summarization task simply because of the nature of the problem as described earlier. Replacing Multi-Layered Perceptron (MLP) in the NSE with an LSTM furthered improved the performance because it remembers what was previously composed and facilitates the composition of novel words. Finally, using memories for each sentence enriches the corresponding word representation, and the document memory enriches the sentence representation.

Table-5 shows the results in comparison to the previous methods. Our hierarchical model outperforms Nallapati et al. (HIER) [4] by 5 ROUGE points. It also performs better than all the existing supervised learning models (non-coverage). All the RL models are initially trained using a supervised learning objective and then further trained in an RL setting. It is observed to boost the

[5] https://pypi.org/project/py-rouge/

ROUGE score by 2-3 points. In the future, we would like to extend our current work in experimenting a few RL models to further improve the performance.

Table **4**: This table shows the progress of the project from a plain NSE model to the Hierarchical model in terms of ROUGE F_1scores on the test set.

Model	ROUGE (% F-score)		
	1	**2**	**L**
• **Plain NSE** ○ The dot-product attention mechanism is too simple.	7.99	0.86	7.52
• **NSE – improved attention** ○ Used a feed-forward neural network to model attention. **[10]**	25.47	8.96	24.01

• **NSE – improved compose**
  ○ Used an LSTM instead of an MLP.
  ○ By remembering past composed words, it enables novel composition at each time step.
	30.86	11.42	29.04

• **Hierarchical NSE**
  ○ Sentence memory enriches word representation.
  ○ Document memory enriches sentence representation.
	37.53	15.69	34.98

Table 5: ROUGE F_1 scores on the test set. Our hierarchical (Hier) and Multi-Head attention models outperform the pointer-generator models. The Big model has 30 sentences and 25 words per sentence in the input document sequence.

Paradigm	Models	ROUGE (% F-score)		
		1	2	L
Supervised learning	HierAttn [4]	32.75	12.21	29.01
	abstractive model [4]	35.46	13.30	32.65
	Pointer-generator [2]	36.44	15.66	33.42
	Hier-NSE (Ours)	37.53	15.69	34.98
	MultiHead-Hier-NSE (Ours)	37.01	15.36	34.74
	Hier-NSE-Big (Ours)	38.12	16.16	35.53
Reinforcement learning	MLE + RL, with intra attention [28]	39.87	15.82	36.90
	DCA, MLE + RL [6]	41.69	19.47	37.92

Sample Outputs

Following are a few sample outputs:

Article:

Aston Villa and Reading have been charged by the Football Association in relation to crowd disturbances occurring after their FA Cup quarter-final matches. It is alleged that Villa and Reading failed to prevent spectators encroaching on the pitch following the clubs ' sixth-round victories against West Brom and Bradford respectively. Both clubs have until Thursday to respond to the charge. Reading fans invaded the pitch after the FA Cup quarter-final replay against Bradford at Madejski Stadium. Aston Villa supporters mobbed Fabian Delph and Co

after they beat West Brom at Villa Park. The Midlands club 's 2-0 win at Villa Park had to be temporarily stopped as fans spilled onto the pitch towards the end of the game and the full-time whistle prompted a mass pitch invasion, as Tim Sherwood 's side booked their place at Wembley. Seats were also thrown from the stand holding West Brom 's supporters and some Baggies players were confronted by celebrating fans. West Brom expressed their ` deep concern ' for the safety of the players after the match and manager Tony Pulis described the scenes as ` disgraceful '. A shirtless man hangs from the crossbar after Villa beat the Baggies in the FA Cup quarter-final Hundreds of Villa fans celebrated victory over their local rivals by storming onto the pitch. Reading fans also invaded the pitch after the Royals beat Bradford 3-0 in a replay at the Madejski Stadium and one supporter entered the pitch during the contest. Reading lost 2-1 after extra-time on Saturday to Arsenal, who will play Villa in the FA Cup final after Sherwood 's side beat Liverpool 2-1. Police formed a line to separate the Reading supporters on the pitch from the Bradford fans in the stands Hundreds of Reading fans surrounded players from both sides after the Royals won a place in the semi-finals.

Ground truth Summary:

Aston Villa and Reading both charged with failing to stop spectators encroaching on to the pitch during fa cup quarter-finals. Villa 's victory over West Brom was marred by chaotic scenes. reading 's win over Bradford saw fans invade the Madejski stadium pitch. Clubs have until Thursday to respond to fa charges.

Hier NSE:

Villa and Reading have been charged by the football association in relation to crowd disturbances. Both clubs have until Thursday to respond to the charge. West Brom expressed their deep concern for the safety of the players.

Figure **10**: (Sample output-1)

Article: The build-up for the blockbuster fight between Floyd Mayweather and Manny Pacquiao in Las Vegas on May 2 steps up a gear on Tuesday night when the American holds an open workout for the media. The session will be streamed live across the world and you can watch it here from 12am UK -LRB- 7pm EDT -RRB-.
Gold Summary: Floyd Mayweather holds an open media workout from 12am uk -lrb- 7pm edt -rrb-. The American takes on Manny Pacquiao in Las Vegas on May 2. Mayweather 's training is being streamed live across the world.
Hier NSE: Floyd Mayweather and Manny Pacquiao in Las Vegas on May 2 steps up. the session will be streamed live across the world and you can watch it here from 12am uk -lrb- 7pm 7pm.

Figure **11**: (Sample output-2)

Article: Blackpool are in talks to sign Austria defender Thomas Piermayr. The 25-year-old has been training with the Championship club this week and they are keen to get him on board for what is expected to be confirmed as a campaign in League One next season. Piermayr is a free agent and had been playing for Colorado Rapids. The former Austria U21 international had a spell with Inverness Caledonian Thistle in 2011. Thomas Piermayr -LRB- left in action for the Colorado Rapids -RRB- tries to tackle Obafemi Martins last year
Gold Summary: Thomas Piermayr has been training with Blackpool this week

Austrian defender is a free agent after leaving MLS side Colorado Rapids Blackpool are bottom of the Championship and look set to be relegated
Hier NSE: Blackpool in talks to sign Austria defender Thomas Piermayr Piermayr has been training with the championship club this week The 25-year-old had been playing for Colorado rapids

Figure **12**: (Sample output-3)

References

[1] A. M. Rush, S. Chopra and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," in *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, pp. 379-389.

[2] A. See, P. J. Liu and C. D. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," in *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, vol. 1, July. 2017, pp. 1073-1083.

[3] R. Nallapati, F. Zhai and B. Zhou, "SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents," in *Proc. of the 31st AAAI Conf. on Artificial Intelligence*, San Francisco, California, USA, Feb. 2017, pp. 3075–3081.

[4] R. Nallapati, B. Zhou, C. N. d. Santos, Ç. Gülçehre and B. Xiang., "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond," in *Proc. of The 20th SIGNLL Conf. on Computational Natural Language Learning*, Berlin, Germany, 2016, pp. 280-290.

[5] R. Meyer, "How Many Stories Do Newspapers Publish Per Day?," 26 May 2016. [Online]. Available: https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/. [Accessed 12 July 2019].

[6] A. Celikyilmaz, A. Bosselut, X. He and Y. Choi., "Deep Communicating Agents for Abstractive Summarization," in *Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana, vol. 1, June. 2018, pp. 1662–1675.

[7] S. Henß, M. Mieskes and I. Gurevych, "A Reinforcement Learning Approach for Adaptive Single- and Multi-Document Summarization," in *Proc. of the Intern. Conf. of the German Society for Computational Linguistics and Language Technology, GSCL*, Germany, 2015, pp. 3-12.

[8] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Proc. of the 27th Intern. Conf. on Neural Information Processing Systems*, Montreal, Canada, vol. 2, 2014, pp. 3104-3112.

[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 28, 2015, pp. 1693-1701.*

[10] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 1 September 2014. [Online]. Available: https://arxiv.org/abs/1409.0473. [Accessed 7 July 2019].

[11] M.-T. Luong, H. Pham and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, Sep. 2015, pp. 1412-1421.

[12] S. Chopra, M. Auli and A. M. Rush, "Abstractive Sentence Sentence Summarization with Attentive Recurrent Neural Networks," in *Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, pp. 93-98.

[13] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, "Hierarchical Attention Networks for Document Classification," in *Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, June. 2016, pp. 1480-1489.

[14] Y.-C. Chen and M. Bansal, "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, vol. 1, July. 2018, pp. 675–686.

[15] J. Ling and A. Rush, "Coarse-to-Fine Attention Models for Document Summarization," in *Proc. of the Workshop on New Frontiers in Summarization*, Copenhagen, Denmark, Sep. 2017, pp. 33–42.

[16] T. Munkhdalai and H. Yu, "Neural Semantic Encoders," in *Proc. of the 15th Conf. of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain, vol. 1, April. 2017, pp. 397–407.

[17] W. T. Hsu, C.-K. Lin, M.-Y. Lee, K. Min, J. Tang and M. Sun, "A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Melbourne, Australia, July. 2018, pp. 132–141.

[18] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult.," *IEEE Transactions on Neural Networks.,* vol. 5, no. 2, pp. 157-166, 1994.

[19] J. Camacho-Collados and M. T. Pilehvar, "From Word to Sense Embeddings: A Survey on Vector Representations of Meaning," *Journal of Artificial Intelligence Research 63*, pp. 743-788, 2018.

[20] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato and T. Mikolov, "DeViSE: A Deep Visual-Semantic Embedding Model," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 2121-2129.

[21] J. Pennington, R. Socher and C. Manning, ""Glove: Global Vectors for Word Representation"," in *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1532-1543.

[22] K. Cho, v. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1724-1734..

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.,* vol. 9, no. 8, pp. 1735-1780, 1997.

[24] K. Cho, B. van Merriënboer, D. Bahdanau and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," in *Proc. of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, Oct. 2014, pp. 103-111.

[25] D. Wang, S. Zhu, T. Li and Y. Gong, "Multi-document Summarization Using Sentence-based Topic Models," in *Pro. of the ACL-IJCNLP 2009 Conf. Short Papers*, Suntec, Singapore, pp. 297-300.

[26] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *In Mining Text Data*, Springer, 2012, p. 43–76.

[27] O. Vinyals, M. Fortunato and N. Jaitly, "Pointer Networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2692-2700.

[28] R. Paulus, C. Xiong and R. Socher, "A Deep Reinforced Model for Abstractive Summarization," 2017. [Online]. Available: http://arxiv.org/abs/1705.04304. [Accessed 12 July 2019].

[29] C.-Y. Lin, ""ROUGE: A Package for Automatic Evaluation of Summaries"," in *Text Summarization Branches Out*, Barcelona, Spain, July. 2004, pp. 74-81, 2004.

[30] S. Banerjee and A. Lavie, ""METEOR: An Automatic Metric for {MT} Evaluation with Improved Correlation with Human Judgments"," in *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan, June. 2005, pp. 65-72.

[31] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation,* vol. 1, no. 2, pp. 270-280, 1989.

[32] M. Ranzato, S. Chopra, M. Auli and W. Zaremba, "Sequence Level Training with Recurrent Neural Networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06732. [Accessed 12 July 2019].

[33] Y. Keneshloo, T. Shi, N. Ramakrishnan and C. K. Reddy, "Deep Reinforcement Learning For Sequence to Sequence Models," Aug 2018. [Online]. Available: http://arxiv.org/abs/1805.09461. [Accessed 19 Jul 2019].

[34] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*, Springer, 1992, pp. 5-32.

[35] S. Narayan, S. B. Cohen and M. Lapata, "Ranking Sentences for Extractive Summarization with Reinforcement Learning," in *Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana, vol. 1, June. 2018, pp. 1747–1759.

[36] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross and V. Goel, "Self-critical sequence training for image captioning," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 1179-1195.

[37] V. Mnih, A. P. Badia, A. G. Mehdi Mirza, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of The 33rd Intern. Conf. on Machine Learning*, New York, New York, USA, vol. 48, June. 2016, pp. 1928–1937.

[38] A. Graves, G. Wayne and I. Danihelka, "Neural Turing Machines," 2014. [Online]. Available: http://arxiv.org/abs/1410.5401. [Accessed 12 July 2019].

[39] J. Weston, S. Chopra and A. Bordes, "Memory Networks," 15 October 2014. [Online]. Available: https://arxiv.org/abs/1410.3916. [Accessed 12 July 2019].

[40] E. Grefenstette, K. M. Hermann, M. Suleyman and P. Blunsom, "Learning to Transduce with Unbounded Memory," in *Proc. of the 28th Intern. Conf. on Neural Information Processing Systems*, Montreal, Canada, vol. 2, 2015, pp. 1828-1836.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998-6008.

[42] S. Takase, J. Suzuki, N. Okazaki, T. Hirao and M. Nagata, "Neural Headline Generation on Abstract Meaning Representation," in *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing*, Austin, Texas, pp. 1054-1059.

[43] B. Sankaran, H. Mi, Y. Al-Onaizan and A. Ittycheriah, "Temporal Attention Model for Neural Machine Translation," 2016. [Online]. Available: http://arxiv.org/abs/1608.02927. [Accessed 12 July 2019].

[44] J. Suzuki and M. Nagata, "RNN-based Encoder-decoder Approach with Word Frequency Estimation.," 2016.

[45] P. Over and J. Yen, "An introduction to DUC 2003 - Intrinsic evaluation of generic news text summarization systems," 2003.

[46] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," in *Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, Pennsylvania, July. 2002, pp. 311-318.

[47] H. Tingting, "In China, 200,000 people die every year because of medication errors, and they are wary of three types of errors.," 17 September 2014. [Online]. Available: https://ln.qq.com/a/20140917/018143.htm. [Accessed 12 July 2019].

[48] Z. Tu, Z. Lu, Y. Liu, X. Liu and H. Li, "Modeling Coverage for Neural Machine Translation," in *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, vol. 1, Aug. 2016, pp. 76-85.