Employing Neural Hierarchical Model with Pointer Generator Networks for Abstractive Text Summarization

by

Wasifa Chowdhury

B.Sc., BRAC University, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY
Fall 2019

Approval

Name:	**Wasifa Chowdhury**
Degree:	**Master of Science (Computing Science)**
Title:	**Employing Neural Hierarchical Model with Pointer Generator Networks for Abstractive Text Summarization**
Examining Committee:	**Chair:** James P. Delgrande Professor
	Fred Popowich Senior Supervisor Professor
	Anoop Sarkar Supervisor Professor
	Oliver Schulte Internal Examiner Professor
Date Defended/Approved:	October 31, 2019

Abstract

As the growth of online data continues, automatic summarization is integral in generating a condensed version of a text while preserving the meaning of the original input. Although most of the earlier works on automatic summarization use extractive approaches to identify the most important information of a document, recent research works focus on the more challenging task of making the summaries abstractive. Sequence-to-sequence models with attention have quantitatively shown to be effective for abstractive summarization, but the quality of the generated summaries is often poor with incorrect and redundant information. In this thesis, we present an end-to-end neural network framework which combines a hierarchical content selector and pointer generator networks abstractor through a multi-level attention mechanism that uses the sentence importance scores from the former model to help the word-level attention of the latter model make better decisions when generating the output words. Hence, words from key sentences will be attended more than words in less salient sentences of input. Our approach is motivated by human writers who tend to focus only on the relevant portions of an article when summarizing while ignoring anything irrelevant that might degrade the output quality. We conduct experiments on the challenging CNN/Daily Mail dataset, which consists of long newswire articles paired with multiple-sentence summaries. Experimental results show that our end-to-end architecture outperforms the extractive systems and strong lead-3 baseline and achieves competitive ROUGE and METEOR scores with previous abstractive systems on the same dataset. Qualitative analysis of test data shows that the generated summaries are fluent as well as informative.

Keywords: automatic abstractive summarization; neural networks; end-to-end framework; natural language generation

"I have made You the polar star of my existence; never again can I lose my way in the voyage of life."

- Rabindranath Tagore, My Polar Star

Acknowledgments

I would like to express my sincere gratitude to a number of people who helped me through my graduate studies. Firstly, I would like to thank my senior supervisor, Dr. Fred Popowich, for his continuous support, patience, and encouragement in the course of this research work. It is because of his guidance and detailed feedback that I was able to complete my thesis. I would also like to convey my thanks to my supervisor Dr. Anoop Sarkar and examiner Dr. Oliver Schulte for their insightful comments. Moreover, I thank Dr. James P. Delgrande to make time to chair my thesis defense.

I am immensely grateful to the nurses and doctors who took care of me during all my medical treatments at Surrey Memorial Hospital and thank Dr. Sonia Cader and Dr. Henry D. Wong for successfully carrying out the surgeries so that I could be in a healthy condition to defend my thesis. Lastly, I would like to acknowledge the love and support of my family and friends, especially Shaika for always being there for me.

Table of Contents

List of Tables

List of Figures

Chapter 1.

Introduction

With more people using the World Wide Web to access information and share ideas with others, there is an influx of user-generated textual content in the form of news, social media, email, forums, etc. To stay up to date amidst this information overload, there is a practical need for automatic summarization tools that provide timely access to, and summary of, various sources. Summarization systems take as input one or more documents and automatically present the most important information as a concise extractive or abstractive summary. While extractive approaches focus on selecting and rearranging key parts of the original text to form a summary, the challenge remains in understanding the semantics of the document and generating a summary that sounds natural and is appropriate for the context or for the task at hand.

1.1. Motivation

More recently, attention-based neural network models [1] have quantitatively shown to be effective for abstractive summarization [10, 11, 59, 60, 63], but the quality of the generated summaries is often poor and pose two problems to be addressed:

Use of a restricted vocabulary: Human-generated summaries include a wide range of words but considering that the number of words in a language, such as English, is very large and that language itself keeps evolving with new words being added to the dictionary, it is challenging for automated systems to cover all the words in a language. Due to computational overhead, these systems use a fixed vocabulary list generally consisting of the most frequent words in the training corpus. This presents a major issue with rare or unknown words that are not part of the predefined list and use a single <*UNK*> token to represent all the out-of-vocabulary (OOV) words. Hence valuable information about the source word is lost, which is a potential reason for making the resulting summaries ambiguous. Table *1.1* compares generated summaries from existing systems with those

written by human summarizers. From the table, we can see that the generated summary from the standard sequence-to-sequence model with an attention mechanism depicts the shortcoming of using a restricted vocabulary. While the human author covers all necessary details including any named entities in the summary, the attentional encoder-decoder framework fails to recover the proper noun, *calbuco*, from the input. Since words like named entities and other factual details (e.g. numbers and dates) occur infrequently in the training data, it is difficult for the system to learn good representations of such rare words. Increasing the vocabulary list to include all unique words in the training set won't solve the problem as the testing data might contain new words, not in the training data, in which case the generated summary would still contain words represented with the <*UNK*> token reducing the performance in test data. Hence there is a need for a different framework in which the system can generate words from the fixed vocabulary distribution and copy any OOV words from the input document to include in the summary.

Lack of informative summaries: An integral problem in the summarization task is how to focus on the relevant parts of an input when generating the output words, especially in case of longer documents that cover many topics. From Table *1.1*, we can see that the final summary is composed of key input sentences (e.g. highlighted portions) and while the human-written summary can convey the main information in a concise way, the existing automated systems struggle to generate summaries that are both readable and informative. The pointer generator networks model [59] is able to handle output readability issues by using a soft switch function to allow the model to learn when to generate a novel word from the fixed vocabulary distribution and when to copy a word from the input, but it struggles to select the most important portions of the source text and instead chooses to summarize secondary pieces of information, as shown by the red portions in the table. As the basic attention mechanism works only at word-level by attending to the input words at each time step of summary generation, the decoder is unable to put more attention on the words from important sentences than those from the less important ones. The need to process the input at multiple levels to extract the right information is supported by both human behavior and previous studies on automatic summarization. For example, when human authors summarize a text, they tend to first identify which sentences contain the necessary information and then use words and phrases from those sentences to paraphrase

while ignoring information from any nonrelevant sentences that might degrade the overall summary quality. Luhn [71], who proposed the seminal paper in automatic summarization in 1958, stated a similar idea that some words in a document are descriptive of its content, and the sentences that convey the most important information in the document are the ones that contain many such descriptive words close to each other. Hence finding the central piece of information requires a better source representation that can encode all necessary contextual information at multiple levels of input.

1.2. Overview of our approach

To overcome these weaknesses of the existing automated systems, which process information mostly at the word-level, we propose to utilize the natural hierarchical structure present in the text, enabling our model to work at different levels of granularity. This research work introduces an end-to-end framework that combines a hierarchical content selector with pointer generator networks [59] and jointly trains the models by minimizing a composite loss function to penalize the disagreement between the generated and correct outputs. The hierarchical selector learns to assign importance scores to input sentences by extracting different semantic features including where a sentence is in the source text, how much information a sentence contains, and how relevant a sentence is with respect to the whole article. Hence, using a combination of these features allows the model to determine which parts of the input contain the most important information. To deal with the problem of OOV words, pointer generator networks are used to learn when to generate a word from vocabulary distribution and when to copy a word from source text to get the final abstractive summary. We combine the selector and abstractor through a multi-level attention mechanism that incorporates sentence-level importance information into the word attention scores to allow the decoder to make better decisions when generating the output words.

While previous works on abstractive summarization used datasets from various genres including scientific and journal articles, social media posts, and Wikipedia pages, we carry out all our experiments on the news domain for two reasons. Firstly, newswire articles are longer in length discussing different topics and the summaries need to be coherent to give

readers a preview of the entire story and help them decide if they want to read the rest of the article. Henceforth due to its length and organization of information, the news genre is suitable for this research work. Secondly, the availability of labeled news corpus, as discussed in detail in Section 4.1 of Chapter 4, makes it easier to compare the performance of our model with previous abstractive models.

1.3. Contribution

To address the problem of automatically generating multiple-sentence summaries from long documents in news corpus, we made several contributions in this thesis which are listed below,

- We propose an end-to-end neural network framework that combines a hierarchical content selector and pointer generator networks-based abstractor through a novel attention mechanism that uses the sentence importance scores from the former model to help the word-level attention of the latter model make better decisions to get the final abstractive summary.

- We conduct experiments on the challenging CNN/Daily Mail dataset [10, 72], which consists of long newswire articles paired with multiple-sentence summaries. Quantitative comparison with ground truth summaries demonstrates the strength of the proposed method. Experimental results show that our end-to-end model outperforms the extractive models and strong lead-3 baseline and achieves competitive ROUGE and METEOR scores with previous abstractive models on the same dataset. Our experimental results also suggest that training the selector and the abstractor jointly by minimizing the overall loss is more effective than training the sub-modules separately.

- Furthermore, to validate the linguistic improvement of the generated summaries, we perform qualitative analysis on test dataset which shows that our end-to-end model can produce summaries which are readable as well as informative. We also provide a visual analysis of the hierarchical attention layer for generated outputs of test set articles in order to demonstrate the effectiveness of the proposed attention mechanism.

1.4. Thesis outline

This thesis is organized into 6 chapters, and the outline of the rest of the chapters is provided below.

Chapter 2 starts by giving an overview of different types of summarization and main stages involved in automatic summarization systems. We then proceed to give an in-depth background on previous abstractive summarization systems, compare the different approaches taken and provide examples of these systems. The chapter ends by explaining the shortcomings of the existing systems and reemphasizes the motivation for this thesis.

Chapter 3 provides details on our proposed architecture. The problem is firstly defined, then the hierarchal content selector and the pointer generator networks abstractor are explained. Finally, details on the two training approaches are provided.

Chapter 4 covers information regarding the dataset used for the experiments and experimental settings of the proposed system.

Chapter 5 elaborates on the experimental results we obtained. The evaluation metrics used to evaluate the performance of the systems are firstly described, then quantitative analysis is performed to compare the scores obtained from our systems with previous summarization models on the same dataset. We also analyze the qualities of the generated summaries and provide examples to show improvement in summary informativeness and relevance. Moreover, the visualization of the proposed attention mechanism for test examples is provided.

Chapter 6 restates the main objectives for this research work and summarizes our findings. We also provide possible avenues for future research.

Table 1.1: Generated summaries for an example from the test set of CNN/Daily Mail corpus showing the problems encountered in existing word-level attentional sequence models. The highlighted input portions contain information used in the reference summary and red portions show secondary input information selected by the system to include in the summary.

Input document (truncated):
(cnn) chile 's calbuco volcano erupted again thursday , marking the third time since last week , the national service of geology and mining said . gregorio billikopf lives across lake llanquihue from the volcano has been photographing and videotaping the three eruptions and described thursday 's event as spectacular but not as severe as the two prior ones . `` there is still smoke on and off , but nothing so dramatic ( as before ) , " said billikopf , a retired university adviser on agricultural issues . `` on a good day i can see about eight volcanoes . `` i understand that the rain that was announced for today would have been a disaster , " he added . he lives in a rainy part of chile , which he described as like a garden of eden . the explosion produced an extensive plume , but it was also described as smaller than the eruptions on april 22 and april 23 , according to cnn chile . deputy interior minister mahmud aleuy said about 1,500 people were evacuated , (….)

Reference:
`` there is still smoke on and off , " says resident with distant view .
the volcano erupts for third time since april 22 .
about 1,500 people are evacuated , an official says , according to cnn chile .

Attentional Encoder-Decoder Model:
UNK volcano erupted again thursday .
UNK volcano has been erupting since last week , the national service of sciences says .
`` there is still smoke on and off , but nothing so big , " he says .

Pointer Generator Networks Model:
chile 's calbuco volcano erupted again thursday .
gregorio billikopf lives across lake llanquihue from the volcano .
he lives in a rainy part of chile , which he described as like a garden of eden .

Chapter 2.

Background

Automatic textual summarization is a broad area, and, in this chapter, we discuss in detail the different distinctions made in the summarization literature, specifically extractive versus abstractive, single versus multi-document, and generic versus query focused. We then proceed to explain the main stages in summarization, then elaborate on what has been done in previous work, and finally discuss the shortcomings of the existing systems and provide additional motivation for our work.

2.1. Automatic Summarization Systems

As defined by Mani [17],

"The goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's need."

While the information source can vary from texts, images, to videos, we restrict our domain to textual data. When generating a summary, different factors come into play–what kind of input and output is used, who the intended audience is, and how the resulting output is presented. Single-document summarization is a widely used task, where systems are fed as input a single document, whether a news story, scientific article or journal, to get output in the form of an abstract, headline, or a simple outline. When multiple documents of the same topic or event are available, multi-document summarization comes in use in order to produce a gist of the content. For example, the content may be a series of news stories on the same event or a set of web pages on some topic or question. In most cases, when deciding what to summarize, only the source document is taken into consideration to get a general summary. But sometimes, a user query is passed as an additional input to summarize a document with respect to any information need expressed in the user query so that a more query focused summarization is achieved. The format of the output also

plays a crucial role in differentiating between various types of summaries. If the goal is only to select and highlight the main points in the source document to make a summary, then extractive summarization is performed. Extractive approaches are extensively used in automatic summarization [8, 18] since the generated summaries tend to be grammatical, fairly readable, and related to the source text, but they are too restrictive in their capability to produce more human-like summaries especially of longer, more complex text like newswire and Wikipedia articles. Writing a concise and fluent abstract requires the capability to reorganize, modify and merge information expressed in different sentences in the input [38]. Hence more research in text generation techniques and abstractive summarization is essential. In Section 2.2, we discuss in detail the different methods used in abstractive summarization.

Before moving on to the next section, it is important to understand the main stages involved in automatic summarization systems, specifically– content selection, information ordering, and sentence realization, as shown in *Figure 2.1* below.

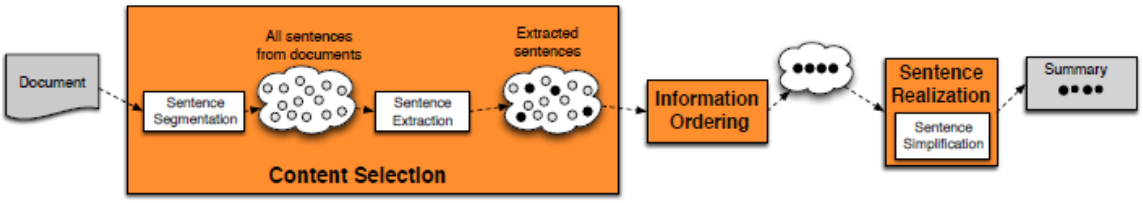

Figure 2.1*:* Basic automatic summarization architecture. Source: [37]

Content selection is a central component in summarization, where the system must determine which information from the input should be included in the output. After selecting the necessary content, the information is organized in the most coherent order through information ordering. Many ordering approaches have been used for better summary readability– from preserving the order in the original document, chronological ordering of the sentences by date, to choosing sentences in which neighboring sentences discuss the same entity. To get the final summary, sentence realization is performed to simplify the extracted sentences and remove anything unnecessary through information fusion or compression. While linguistic rule-based and statistical-based systems generally

handle the tasks separately, in more advanced neural network-based models, all the tasks are usually jointly learned to generate the resulting summary.

2.2. Related Work

In order to develop automated systems that emulate human summarization, a better understanding of the input is required, with sophisticated analysis and semantic interpretation of the text. Different abstractive methods have been used to tackle this issue in several text genres including– emails, newswire stories, scientific and journal articles, social media posts, Wikipedia pages, and narratives. Particularly, the news genre is of interest to researchers, especially for single-document summarization, due to its length and structure. For ease of understanding, the various methods used in this domain can be divided into several categories: (1) linguistic rule-based, (2) statistical models, (3) hybrid, and (4) artificial neural networks and deep learning models. We explain each of these categories in the following subsections.

2.2.1. Linguistic rule-based methods

Many of the earlier works on abstractive summarization use a select and compress approach to include summary-worthy information from the input while discarding any extraneous detail using syntactic and discourse knowledge. Syntactic knowledge makes use of external knowledge sources and extraction rules and then represents the information in alternative structures like templates, trees, ontologies, and graphs. On the other hand, discourse information utilizes the relation between these textual constituents to improve the linguistic quality of the output.

Jing [19] develops a sentence reduction system using editing operations on the extracted sentences to get concise and coherent summaries where 81.3% of the reduction decisions made by the system were consistent with those of humans. In order to make the reduction decisions, the system relies on multiple sources of knowledge including syntactic and contextual information as well as statistics computed from a news corpus consisting of original sentences and their corresponding reduced forms written by human professionals. The reduction algorithm starts by parsing the input sentences into parse tree

representations. Each node of a parse tree is then visited and using additional resources like linguistic rules and a lexicon, the system decides which portions of a sentence can be removed without affecting grammaticality. For example, in an English sentence, the main verb, subject and object are considered essential while a prepositional phrase is not, and a noun phrase remains grammatically correct even if the adjective modifier is dropped. The next step uses context information to identify the sentence phrases most related to the overall topic of the document. To achieve that, each word in the input sentence is assigned an importance score, based on the number of links it has with other source words and the type of links. A link between words is given if they are repetitions, morphologically related, or have lexical relations through Wordnet [13]. With the help of a corpus of human written reductions, the system then computes probabilities of each phrase in the input being removed, reduced and unchanged. The final decision of removing a phrase or subtree from the parse tree is based on all the factors from the previous steps. *Table 2.1* shows an example output of this method.

Table 2.1 Example output of [19]. The italicized portion from the input is selected to include in the summary

Original Sentence: When it arrives sometime next year in new TV sets, *the V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.*
Reduction system: The V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.
Expert: The V-chip will give parents a device to block out programs they don't want their children to see.

Hedge Trimmer [20] is a similar system which uses linguistically motivated heuristics, but without using information about discourse connections and compressing the first sentence of news stories to generate headline-style summaries of only 10 words. Since headlines are usually shorter than sentences and require multiple levels of compression, manual analysis of a large corpus of article-headline pairs was performed to determine the reduction rules. Every input sentence is shortened by choosing the correct noun and verb phrases, removing low content units like determiners and time expressions, and iteratively shortening the sentence by removing constituents until a length threshold is reached. The authors mention

that output generated from a rule-based system is more likely to be grammatically correct but semantically nonsensical, as depicted in *Table 2.2*.

Table 2.2*:* A semantically incorrect output of Hedge Trimmer [20]

HedgeTr: It may not be everyone's idea especially coming on heels

Cheung and Penn [21] move beyond deletion-based compression and reduction techniques and introduce sentence fusion to select several input sentences conveying the most important content and merge them into a single sentence summary. They use an unsupervised clustering algorithm to firstly identify clusters of similar sentences and rank the clusters according to their salience by computing the sum of importance scores of the set of terms of the sentences in a cluster using the signature term method of [22]. The cluster of sentences with the highest score is selected and their dependency tree representations are then fused into an intermediate sentence graph. The graph is further expanded with dependency trees of all other sentences in the source text using different heuristics and an event coreference module is used to ensure the merging of information from different parts in the source text doesn't result in invalid output. An output dependency tree of the expanded sentence graph is generated using linear integer programming and a linearization method is used to get the final sequence of words, considering the ordering of information from the input text. All the experiments were conducted on the TAC 2010 and TAC 2011 Guided Summarization Corpus [23] and their system outperforms previous sentence compression and fusion models, noting the effectiveness of using the entire source text through graph expansion and event coreference resolution algorithms.

While linguistic rule-based models circumvent the need for large amounts of labeled training data, development of the set of rules requires tedious manual analysis and sometimes heavy reliance on external knowledge sources which may not be available for low-resource languages.

2.2.2. Statistical models

Several probabilistic and machine learning models for sequential data are developed to avoid the need to provide linguistic rules as input to the systems by performing statistical analysis on a corpus to combine and weight different features of sentence importance.

Inspired by works on other natural language processing applications like statistical machine translation [24, 25], some research works consider summarization as a problem analogous to machine translation of translating from a verbose to a succinct language. Unlike machine translation, the generated output is typically shorter than the input document length and there is no need for one-to-one word-level alignment between source and target. To generate a sequence of output words, *S*, given an input document *D*, a noisy channel model is used which uses Bayes rule to decompose *P(S|D)* to,

$$P(S|D) = \frac{P(D|S)\,P(S)}{P(D)} \tag{2.1}$$

The idea is that the verbose language, or the observed document, is treated as a garbled version of a concise language, or unseen summary, being transmitted through a noisy channel, and the goal is to find the summary most likely to have been generated given with the observed document. The model is a product of two factors- (1) a conditional summarization model, $P(D|S)$, which computes the conditional probability of the document given a summary, and (2) a language model, $P(S)$, which calculates the probability of a sequence of output words. The factor, $P(D)$, in the denominator can be omitted since the probability of a given document stays constant for every candidate summary. Hence, the most probable summary, $\hat{S}$, is calculated using the following equation,

$$\hat{S} = \arg\max_{S} P(D|S)\,P(S) \tag{2.2}$$

For example, the work of Banko et al. [26] on headline generation from newswire stories uses independently estimated statistical models of content selection, reordering, and surface realization so that the summary candidates are ranked against each other as a

weighted combination of log probabilities of the individual sub-models using the following equation:

$$\widehat{H} = \underset{H}{\arg max} \left(\alpha \sum_{i=1}^{n} \log\big(P(w_i \in H \mid w_i \in D)\big) + \beta \log\big(P(len(H) = n)\big) + \gamma \sum_{i=1}^{n} \log\big(P(w_i \mid w_{i-1})\big) \right) \quad (2.3)$$

$$\text{where } P(w_i \in H \mid w_i \in D) = P(w_i \in D \mid w_i \in H)\, P(w_i \in H)$$

That is, for every input document *D*, the overall probability of a candidate summary, *H*, consisting of words (w_1, w_2, w_3, …, w_n), is computed by modeling the probability of the terms selected for the summary, length of the summary, and the most likely output word sequence, with the weights α, β, and γ learned through cross-validation. To simplify parameter estimation for the content selection model, $P(w_i \in H \mid w_i \in D)$, the conditional probability is estimated using a count-based noisy channel model where the likelihood of a word in a summary is independent of other words in the summary. During surface realization, the probability of a target word sequence is approximated using a bigram model, which computes the probability of the current word by looking at a single preceding word and apply back-off weights [36] for bigrams not seen in the training data. Finally, the Viterbi beam search [27] is used to find the sequence of words, $\widehat{H}$, that maximizes the total probability. The model is also extended to consider additional information like part of speech (POS) and positional information. The systems were trained on around 25,000 labeled news article-headline pairs from Reuters while evaluated on an unseen set of 1000 Reuters articles. While POS information alone doesn't improve on the simpler model, adding positional information helps and the model performance improves when used in combination. The authors point out that in order to generate more coherent summaries, a much larger training corpus is needed, and longer *n*-gram models might lead to better language generation since long-range dependencies would be considered.

HMM Hedge [28] is a system that uses a generative model incorporating Hidden Markov Models (HMMs). The model uses a bigram language model like Banko et al. [26] but trained on a larger training corpus from TIPSTER [29]. The researchers adopt the

additional constraint that the headline words are chosen from the first *N* words of the story in the order that they appear, where *N* is intuitively set at 60. Since headlines are usually written in the present tense while stories are written in the past tense, each story word is also expanded into its set of morphological variants to get the final summaries. To select the most likely headline for an input story, Viterbi algorithm is used with four additional decoding parameters: a length penalty to bias the algorithm towards a specified output length range, a position penalty to favor headline words that occur early in the story, a string penalty to generate headlines from strings of contiguous story words, and a gap penalty to bias against contiguous headline words coming from widely separated story words. Upon evaluation, while the decoding parameters has a positive impact on the results, adding morphological variants has a slight drop in the performance of the model. Like Banko et al. [26], the authors also agree to use trigrams or longer n-gram language models to get fluent headlines.

Statistical-based methods allow the models to learn the relationship between multiple features in source documents and the corresponding features in output summaries but coming up with the hand-crafted features and tuning their weights to optimize the model can be time-consuming and computationally expensive.

2.2.3. Hybrid models

Some works on abstractive summarization address the limitations of individual models by using hybrid models to combine the strengths of different lexical, statistical, and semantic knowledge representation types.

The Topiary system [73] is one such hybrid model which shows its effectiveness by using a combination of linguistically motivated sentence compression technique with statistically selected topic terms to generate headlines from news articles. The system utilizes the Hedge Trimmer [20] discussed earlier to compress incoming sentences and Unsupervised Topic Discovery (UTD) algorithm to add important article topics missing in the compressed output without loss of meaning. UTD is a statistical method which takes as input a large unlabeled document corpus and automatically derives a set of topic models from the corpus, assigning meaningful names to those topic models and using them to find

the most likely topics for each document. The authors address that one problem of using the Hedge Trimmer alone is that it is constrained to produce headlines using only the first sentence from input documents, but a single sentence generally doesn't contain all the important information in a story and in most cases the information is distributed over two or three sentences. They also note that using a topic list alone is a problem as the topics show what the subjects are but rarely give an indication of what the main verbs or events are in the input document. To overcome these drawbacks, Topiary is a modification of the Hedge Trimmer taking a list of topics with relevance scores as additional input so that an informative headline is generated with a broader coverage of the main concepts. Topiary resulted in giving a state-of-the-art performance for single document summarization tasks on the 2004 Document Understanding Conference (DUC), which is a standardized workshop for abstractive summarization using automatic evaluation metrics like ROUGE [30] and BLEU [31] to evaluate the performance of the summarization systems against human-generated summaries. The hybrid model performed much better than either Hedge Trimmer or UTD alone, producing fluent headline-style summaries with additional context, as shown in the following example in *Table 2.3*.

Table 2.3*:* Generated headlines of UTD, Hedge Trimmer [20], and Topiary [73] of an example story about the investigation into the bombing of the U.S. Embassy in Nairobi on August 7, 1998.

UTD: BIN LADEN EMBASSY BOMBING POLICE OFFICIALS PRISON HOUSE FIRE KABILA
HedgeTr: FBI agents this week began questioning relatives of the victims
Topiary: BIN LADEN EMBASSY BOMBING: FBI agents this week began questioning relatives

In some later work, semantic representations like Abstract Meaning Representation (AMR) [32] is introduced to the abstractive summarization task to capture the meaning of a text by giving a specific meaning representation to it. Liu et al. [33] present a novel data-driven graph-to-graph framework using AMR. The input sentences of a source text are firstly parsed to individual AMR graphs using JAMR [34], which is an automatic semantic parser trained on a 20,341-sentence corpus annotated with AMR by human experts. The nodes of an AMR graph are labeled with concepts of a sentence while edges are labeled with the

corresponding semantic relations between the concepts. Hence, sentences that have the same underlying meaning are represented by the same AMR graph. For example, *She likes apple* and *Apples are liked by her* are assigned the same AMR. The AMR graphs of these sentences are then transformed and merged into a single summary AMR graph, which captures the most salient semantic content from the source, through two stages- source graph construction and subgraph prediction. A source graph is constructed by merging the identical concepts in all the AMR graphs while a summary subgraph is selected from the source graph using Integer Linear Programming (ILP) where the coefficients are updated through supervised learning. The final stage of generating a summary text from the summary subgraph is left for future work by the authors due to the unavailability of an AMR-to-text generator at that time. Dohare et al. [35] propose a full-fledged pipeline able to produce the summary text. Unlike Liu et al., they use multiple summary subgraphs, each focusing on information in a different part of the input story, to get the textual output.

Much of the research work on abstractive summarization discussed so far focus only on generating short headline-style summaries, and there has not been much success in producing longer summaries consisting of multiple sentences which is more helpful in providing the user with a detail outline of the main information in a document. With the advent of artificial neural networks (ANNs) and deep learning models, more emphasis is now being put on making use of information in arbitrarily long input sequences and producing multiple-sentence summaries. In the next sections, we talk about the existing neural network-based summarization models and our motivation for this thesis.

2.2.4. Artificial neural networks and deep learning models

ANNs are based on the structure of neurons inside a human brain, which can solve complex problems through the activation of certain connections in the interconnected network of neurons. An ANN is composed of connected layers of nodes, where a single *k*th node takes multiple weighted inputs, $x_j w_{kj}$, and applies an activation function, $\varphi(\cdot)$, to the summation of m inputs, $v_k = \sum_{j=0}^{m} x_j w_{kj}$, to get the desired output, $y_k = \varphi(v_k)$, as illustrated in Figure 2.2.

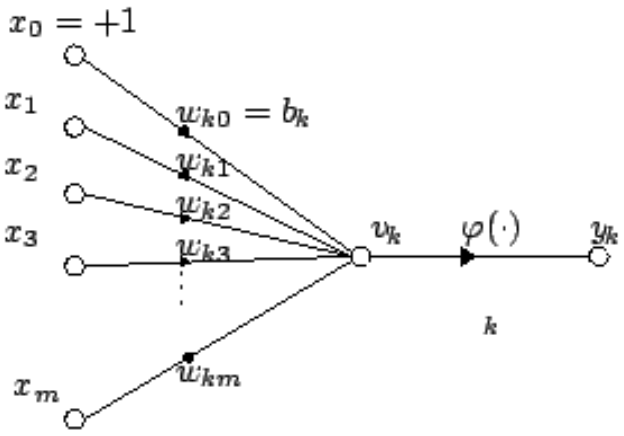

Figure 2.2*:* Basic structure of a single node. Source: [39]

In the above diagram, the node weighs up different factors in the input to make the output decision, where a bias term, b_k, is added as a threshold for the node to be activated, and the activation function allows the node to learn nonlinear features of the input along with linear features.

The simplest ANN is a feed-forward neural network consisting of 3 layers- input, output and one hidden layer, where each node in one layer has directed connections to the nodes of the subsequent layer to allow information to be passed in the forward direction only. With networks containing multiple hidden layers, also called a multi-layer perceptron (MLP), more complex and abstract features of the input are learned to facilitate in the final output decision. The output is calculated from the input through a forward propagation, where the output from the previous layer is fed as input to the next layer, while the weights and biases are adjusted to minimize the difference between the actual and calculated outputs through backpropagation [42] and different optimization algorithms. Figure 2.3 shows the basic architecture of a neural network with multiple hidden layers.

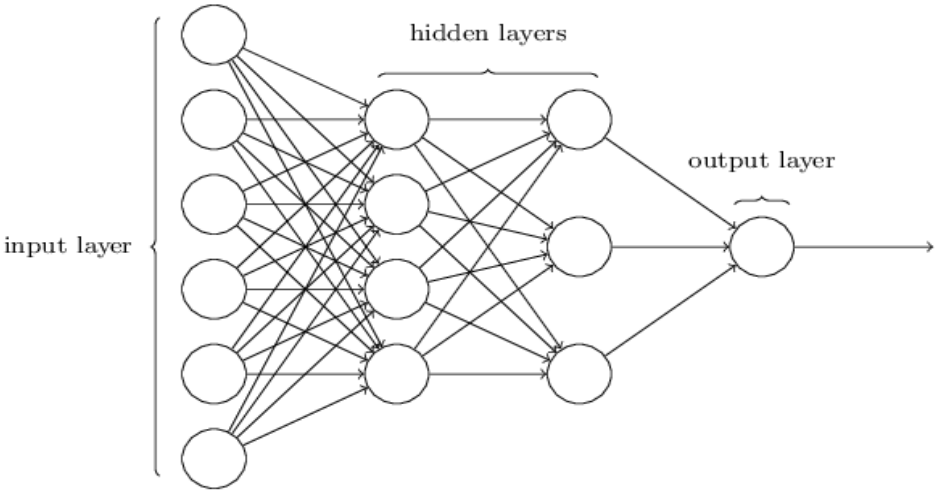

Figure 2.3*:* Feed-forward neural network with multiple hidden layers. Source: [40]

A major shortcoming of feed-forward neural networks is their inability to capture information from previous computations in the time since nodes at each layer have separate weights and biases which behave independently. Recurrent Neural Networks (RNNs) are designed to be suitable for sequential and temporal data, where information about what has come previously is needed to make the next prediction. As shown in Figure 2.4, RNNs have feedback loops which store information about previous context in a sequence of timesteps. At each time step *t*, x_t is fed as input and the hidden state, s_t, is calculated not only from the current input but also based on the previous hidden state, s_{t-1}. Hence, $s_t = f(W[s_{t-1}, x_t] + b)$ where $f(\cdot)$ is a nonlinearity function and the output at that step is, $o_t = \varphi(Vs_t)$. Here, $\varphi(\cdot)$ is the activation function while *W*, *V*, and b are learnable parameters shared across all timesteps which greatly reduces the total number of parameters the model learns. Note that $[s_{t-1}, x_t]$ is a matrix concatenating the hidden state and given input.

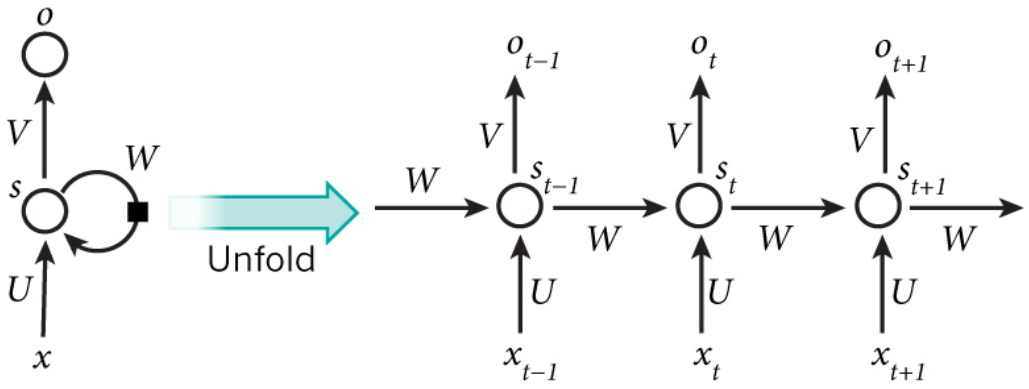

Figure 2.4*:* Unfolded basic recurrent neural network. Source: [41]

While vanilla RNNs can look back at previous time steps, they have difficulty capturing long term dependencies between steps that are far apart due to vanishing and exploding gradient problem when trained with backpropagation [46]. Hence Hochreiter et al. [7] introduce a variant of RNN, known as long short-term memory (LSTM), which uses gate-like structures to remove and add information to an intermediate memory called cell state. The hidden state at each time step, *t*, is now computed using the following equations,

$$\begin{aligned} f_t &= \sigma\big(W_f[s_{t-1}, x_t] + b_f\big) \\ i_t &= \sigma(W_i[s_{t-1}, x_t] + b_i) \\ \widetilde{C_t} &= tanh(W_C[s_{t-1}, x_t] + b_C) \\ C_t &= f_t C_{t-1} + i_t \widetilde{C_t} \\ out_t &= \sigma(W_o[s_{t-1}, x_t] + b_o) \\ s_t &= out_t \, tanh(C_t) \end{aligned} \tag{2.4}$$

Here, f_t is the forget gate function, which takes the previous hidden state, s_{t-1}, and current input token, x_t, and pass them through a *sigmoid* layer to decide what information to throw away from the cell state. After that, new information is added to the cell state using an input gate layer, i_t, which decides on the values to be updated, and another layer, $\widetilde{C_t}$, to decide on the new candidate values to be added. The new cell state, C_t, is computed by updating the old cell state, C_{t-1}, with this information. Finally, the hidden state, s_t, is calculated by finding which parts of the cell state to output using the output layer, out_t. Over time, other versions of RNN and LSTM are presented such as peephole LSTMs [47], Gated Recurrent Units (GRU) [43], and Depth Gated RNNs [48].

Different neural network architectures are further introduced to serve various purposes like- Convolutional Neural Networks (CNNs) which considers the spatial structure of data more commonly used in image analysis, recursive neural networks used in case of structured input, unsupervised neural networks such as autoencoders for learning compact data representations, and generative models such as variational autoencoders and Generative Adversarial Networks (GANs) which can learn to mimic any distribution of data. Moreover, deep learning models with multiple layers of varying sizes between input and output layers are used to learn complex non-linear relationships.

Motivated by the success of using a sequence-to-sequence model in machine translation [1, 43], Rush et al. [12] apply ANN to abstractive summarization and show significant performance gains over previous non-neural network-based models. Sequence-to-sequence models are used to map a sequence of symbols from an input sequence to an output sequence using an encoder-decoder architecture with two separate ANN models. On the encoder side, an input sequence is fed to a neural network model to get a fixed-length vector capturing information for all input elements and on the decoder side, the encoded representation is fed to another neural network model to generate the output sequence. Rush et al. experiment on three different encoders: 1) a simple bag-of-words encoder which embeds the words of the input sequence to an embedding size, ignoring properties of order or structure of the words, 2) a convolutional encoder which uses CNN to model local interactions between words, and 3) an attention-based encoder which learns a latent soft alignment over the input sequence to help inform the output summary. The attention mechanism addresses the problem of encoding all input information into a single vector by allowing the model to learn what to attend to as a function of the input text and what it has produced so far. A feed-forward neural network language model, like [44], is then used as a decoder to estimate the contextual probabilities of words in the output vocabulary and the beam search algorithm generates the most likely sequence of words. Unlike the statistical noisy channel approach which splits and independently estimates a language model and a conditional summarization model, both the encoder and decoder are trained jointly on the sentence summarization task by estimating the parameters to minimize the negative log-likelihood of a set of article-summary pairs using mini-batch stochastic gradient descent. An additional log-linear model is utilized to include rare input features,

like proper nouns, names, and places, in the output text. The models are trained and evaluated on the DUC 2003 and 2004 tasks as well as a large-scale dataset of Gigaword [45] consisting of millions of examples from different newswire sources. The authors note that one of the shortcomings of the models is making factual errors and reordering words in the wrong way. *Table 2.4* shows one such instance where the main subject is wrong.

Table 2.4*:* Example from [12]. I is the input, G is the true output, A is the seq-to-seq abstractive model, and A+ is the seq-to-seq abstractive model with additional extractive features.

I: the white house on thursday warned iran of possible new sanctions after the un nuclear watchdog reported that tehran had begun sensitive nuclear work at a key site in defiance of un resolutions . **G**: us warns iran of step backward on nuclear issue **A**: iran warns of possible new sanctions on nuclear work **A+**: un nuclear watchdog warns iran of possible new sanctions

Chopra et al. [11] improve performance on the same datasets by using an RNN model as decoder instead of a feed-forward neural language model. *Figure 2.5* shows the basic components of this standard sequence-to-sequence model with attention. In the diagram, the encoder RNN reads the input text word-by-word to produce a sequence of encoder hidden states. The decoder RNN then updates the decoder hidden state using the previous hidden state and current input at each step. The attention distribution is computed using the updated decoder state and encoder hidden states. The attention distribution and the encoder states are used as a weighted sum to get the context vector, which informs the decoder on what has been read so far from the input text. Finally, the context vector and the updated decoder state are used to get a vocabulary distribution over all the words in the predefined list and the word with the highest probability is predicted as the next word. In the diagram, the <*START*> token on the first step is used as a signal for the decoder to start generating, and we can see that the decoder has produced *Germany* so far and is attending to the input words *win* and *victorious* to predict the next word as *beat*. Other works which propose extensions to this standard architecture, for further performance gains include: 1) Nallapati et al. [49] who explore a large vocabulary trick (LVT) described in [52] to reduce the size of the output vocabulary by including words of only the source documents in a batch and a small set of most frequent target words, 2) Nallapati et al. [10] propose several novel

models, such as, a) a model with feature-rich encoder is used to capture additional linguistic features of the input like POS tags, named entities, and term frequency (TF) and inverse document frequency (IDF) values of the words, and b) another model with a hierarchical attention is used to jointly model attention distribution at both words and sentence levels, 3) Takase et al. [50] incorporate information present in AMRs such as predicate-argument structures as additional semantic features, and 4) Miao et al. [51] augment the baseline model with variational autoencoders.

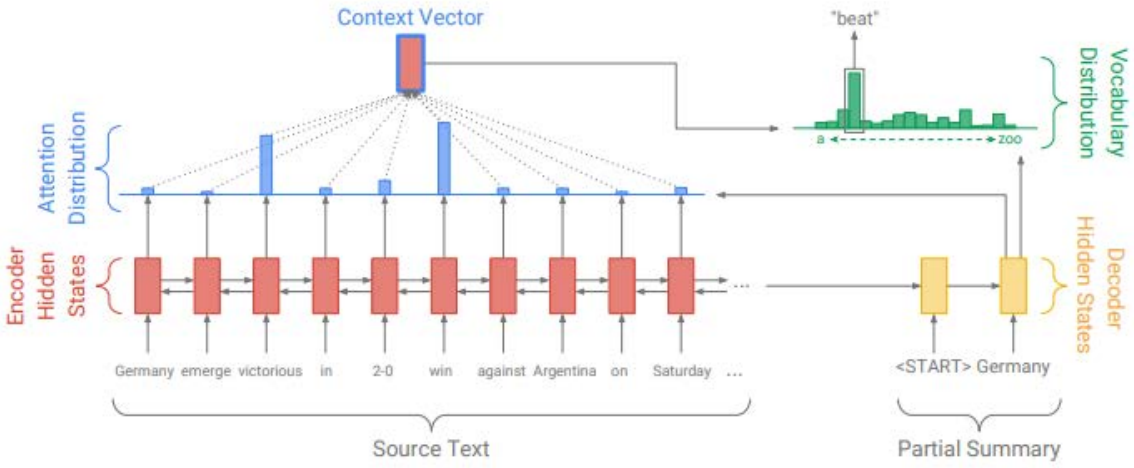

Figure 2.5*:* Baseline sequence-to-sequence model with attention. Most of the later abstractive summarization models are extensions to this standard model. Source: [59]

In textual summarization, since it is natural to reuse some words like named-entities and proper nouns, from the original text in the summarized text, some of the research work uses the concept of pointer networks [53] to correctly reproduce factual details by allowing the system to copy words from the source to output text. The intuition behind pointer networks comes from psychological evidence that humans naturally tend to point towards objects in the context or the environment when the name of an object is not known [54, 55]. Gu et al. [56] introduce a COPYNET model that incorporates pointer networks into an encoder-decoder structure for abstractive summarization by using a shared *softmax* function at each decoder timestep to decide whether to generate a word from the output vocabulary or copy a word from the input document. In a similar work, Nallapati et al. [10] and Gulcehre et al. [57] use models consisting of two *softmax* layers to predict the next word in the decoder-

one is used to predict the location of a word in the source sentence, and the other is used to predict a word from the target vocabulary. The decision of which *softmax* layer to use at each decoder timestep is made using a switching probability which is computed using MLP with *sigmoid* output function. Inspired by human readers, who read a text multiple times before writing a summary, Zeng et al. [58] introduce a mechanism to let the encoder read the source document in two passes and use the information in the first read to bias the second read representation to capture overall meaning of the document. Unlike [10, 56, 57], they use a copy mechanism with a smaller vocabulary to handle OOV words. See et al. [59] introduce pointer generator networks which is one of the most successful models using pointer networks. Pointer generator networks can automatically learn to choose to point to an input word or generate a vocabulary word by using an extended vocabulary which combines the probabilities from the attention and predefined vocabulary distributions, unlike [10, 56, 57, 58] which do not mix the probabilities from the distributions. The additional coverage mechanism restrains their model to attend to the same input positions which reduce repeated phrases from appearing when generating multiple sentence summaries. *Table 2.5* shows a comparison of outputs from the baseline sequence-to-sequence attentional model and the pointer generator networks with coverage models on the same input. The former model represents all OOV words, like *muhammadu buhari*, with a single <*UNK*> token while the latter model correctly reproduces the proper name. Other related works include: Hsu et al. [16] who use a unified model where the word-level attention is rescaled by the sentence-level attention with an additional sub-loss function to penalize inconsistency between the two levels of attention, and Li et al. [60] who propose a key information guide network to obtain keywords from input text to help in the generation of output text.

Table 2.5*:* Comparison of output from baseline seq-to-seq attentional model and pointer generator network with coverage model of [59] on the same input.

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, *muhammadu buhari* told cnn's christiane amanpour that he plans to aggressively fight corruption that has long plagued nigeria and go after the root of the nation's unrest. buhari said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, he said his administration is confident it will be able to thwart criminals and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. buhari defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.
Baseline Seq-to-Seq + Attention: UNK UNK says his administration is confident it will be able to destabilize nigeria's economy. UNK says his administration is confident it will be able to thwart criminals and other nigerians. he says the country has long nigeria and nigeria's economy.
Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

While these models learn by minimizing the maximum likelihood loss on the set of article-summary pairs in a supervised way by providing the correct output word as input at each decoder timestep during training, many recent studies use a new way of training summarization models via reinforcement-learning based methods [61], where the models are trained with a reward signal that depends on the entire output sequence and the reference summary. Aurelio et al. [62] propose a hybrid loss function, known as Mixed Incremental Cross-Entropy Reinforce (MIXER), which combines both cross-entropy and reinforcement-based learning, to mitigate the discrepancy between the optimization objective and automatic evaluation metrics, such as ROUGE, by directly optimizing evaluation metrics. Paulus et al. [63] and Kryscinski et al. [64] also use a mixed training objective but with a self-critical policy gradient algorithm [65]. Liu et al. [66] utilize GAN where an additional discriminator model can distinguish between natural and generated summaries to guide the generative model towards outputs more like human-written text. Pasunuru et al. [67] add novel reward functions to include more salient phrases of input in summary and a multi-reward optimization approach to optimize these multiple rewards simultaneously with reinforcement learning, while Chen et al. [68] introduce a novel sentence-level policy gradient method which allows their model to compress extracted

sentences to produce a concise summary. Reinforcement-based learning models can help increase quantitative scores by optimizing a discrete metric like ROUGE but doesn't guarantee an increase in the quality and readability of the output. Moreover, training the models solely with reinforcement-based learning can be slow and difficult to tune, while a mixed objective might help reduce the training time.

2.3. Discussion

This chapter provided an in-depth background on the different approaches to automatic abstractive summarization from using linguistic rule-based and statistical methods to artificial neural networks. Although neural networks-based techniques have made generation of multiple-sentence summaries from longer textual documents viable, most of the existing models don't focus on what content to summarize from the source text, for which reason, these abstractive models can produce fluent summaries but the quality of the generated summaries is often poor with incorrect and redundant information. As shown in Table *1.1*, the summaries are composed of the most important information from the articles, thus a summarization model should be able to successfully perform the content selection. Hence, in this research work, we are interested in taking advantage of the hierarchical structure of text to identify salient content from the input at multiple levels and summarize the salient content in an abstractive way. In the next chapter, we propose an approach which combines a hierarchical content selector with pointer generator networks to generate abstractive summaries that are better at capturing the aboutness of the input documents.

Chapter 3.

Hierarchical Model with Pointer Generator Networks

Figure 3.1 shows the block diagram of our proposed end-to-end model comprising of the main components of a hierarchical content selector and a pointer generator networks abstractor connected through a multi-level attention mechanism. Nallapati et al. [10] and Hsu et al. [16] offer similar models to ours but differ in the way the attention is computed to link the intermediate submodules. Figure *3.2* gives an overview of the complete model architecture which will be explained in the following sections. We firstly begin by defining our problem in Section 3.1, then proceed to explain the subcomponents of the model in sections 3.2 and 3.3, and finally provide information on training the models in Section 3.4.

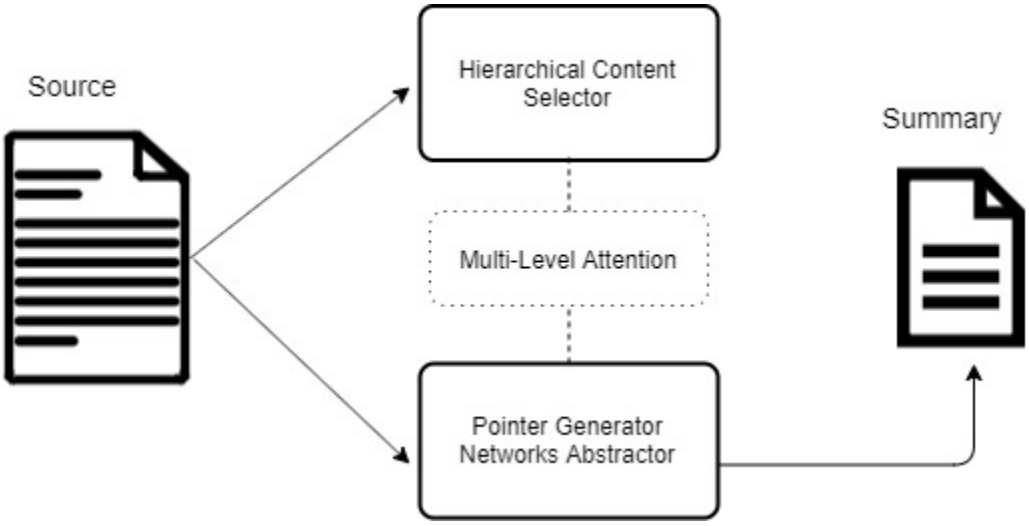

Figure 3.1*:* Block diagram of our end-to-end model, where the hierarchical content selector and pointer generator networks abstractor is combined through a multi-level attention mechanism and jointly trained.

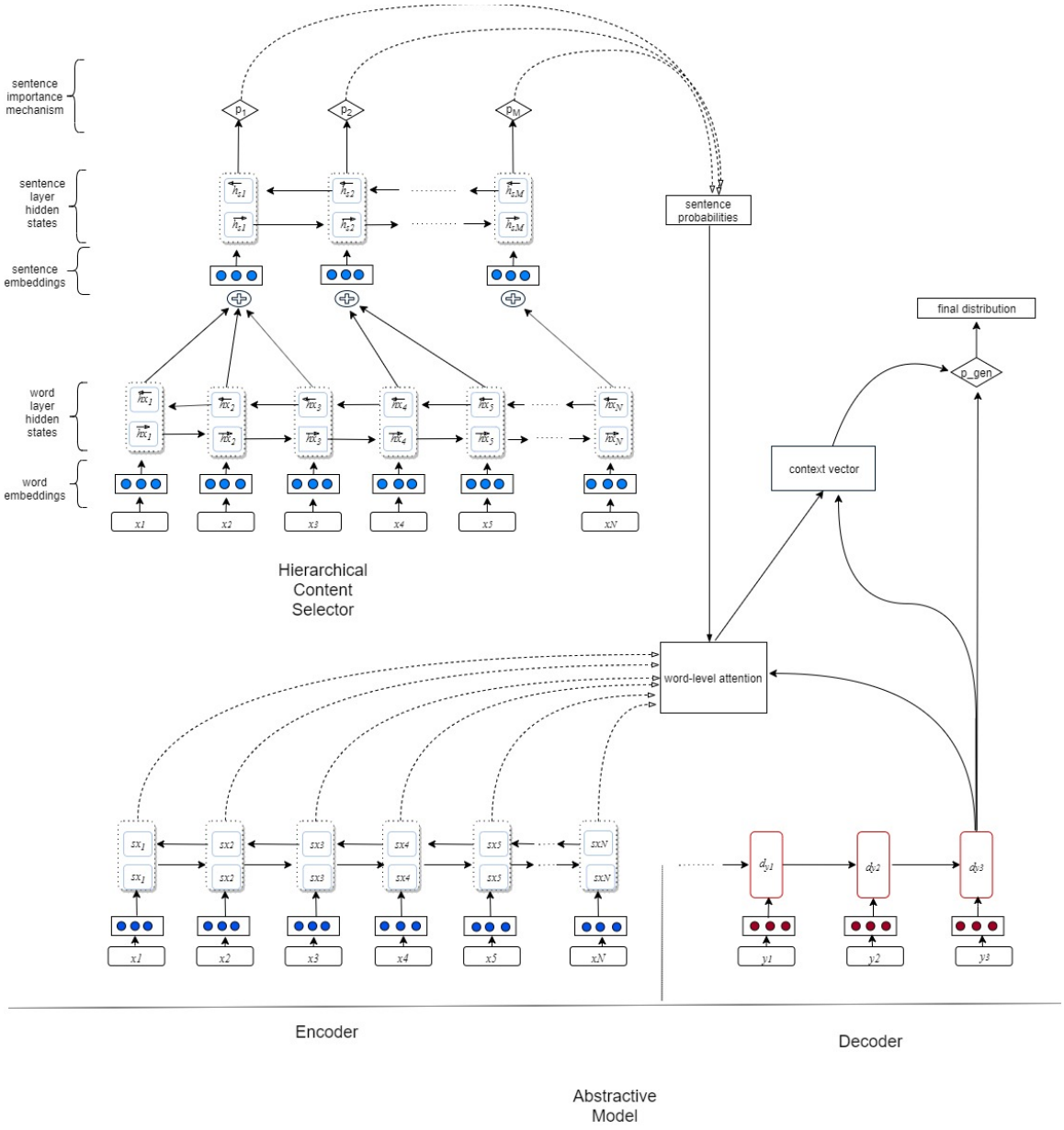

Figure 3.2: Proposed complete architecture, where the blue boxes represent encoder hidden states and red boxes represent decoder hidden states. The diamond-shaped boxes indicate sigmoid function and the arrows show the information flow.

3.1. Problem Definition

For the problem of automatic summarization, given an input x, the objective is to select $\acute{y}$ from the candidate output sequences by modeling $\acute{y} = argmax_y \log P(y|x)$. We can define y as a summary consisting of a sequence of L words, such that $y = y_1, \ldots\ldots, y_L$ with $L < |x|$, and each y_t represents a word in the target-side vocabulary of size $|V_y|$. Our input x is a

piece of text corresponding to an online news article and unlike other previous abstractive models where the encoder works only at word-level, our abstractive model utilizes an intermediate hierarchical content selector to extract the key concepts at both sentence and word levels before generating the final summary. Let x comprise of N words where each word x_k is part of a source-side vocabulary of size $|V_x|$ and is represented as $x = x_1, x_2, \ldots\ldots, x_N$. Every x_k also belongs to a sentence from M sentences in the article, with *index(k)* mapping the positional index of the sentence to which k^{th} word belongs. All the words are mapped to vector representations by looking up in a matrix of size V x D_w, where $V = |V_x| + |V_y|$ and D_w is the size of the word embeddings. We represent the resulting vectors as $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots\ldots, \boldsymbol{x_N}$ and $\boldsymbol{y_1}, \ldots\ldots, \boldsymbol{y_L}$. Our proposed end-to-end framework to compute the log conditional probability of the sequential textual data is described in the next sections.

3.2. Hierarchical Content Selector

Our insight behind using an intermediate hierarchical content selector is based on human summarizers who tend to focus only on the relevant portions of an article including any key sentences and words while ignoring anything irrelevant that might degrade the quality of the generated summary. As RNNs have shown great promise in many NLP tasks (e.g. language modeling [3], machine translation [1]) in its ability to model sequential information of variable length, we incorporate two bidirectional RNNs, one at the word level and the other at the sentence level to compute the hidden states and use a mechanism to weigh different parts of an article based on their importance features.

3.2.1. Word-level Representation

Each input article x is tokenized into words and passed one at a time through a one-layer bidirectional LSTM network. We compute the forward representation at each time step i by feeding in the current word embedding and previous hidden state, and in a similar way calculate the reverse representation using the word and its right context. To get a better representation of the input text, we concatenate both the states to get the final hidden state of each word. Note that at the first step $i=1$, the previous hidden state is initialized to zero.

$$\begin{aligned} \vec{h}_i &= \overrightarrow{LSTM}\left(\boldsymbol{x_i}, \vec{h}_i^{prev}\right) \\ \overleftarrow{h}_i &= \overleftarrow{LSTM}\left(\boldsymbol{x_i}, \overleftarrow{h}_i^{prev}\right) \\ h_i &= \left[\vec{h}_i, \overleftarrow{h}_i\right], i \in [1, N] \end{aligned} \tag{3.1}$$

3.2.2. Sentence-level Representation

Since a document consists of many sentences each of which is made up of many words, we find the sentence embedding by calculating the average over all the hidden states of the words in a sentence. We then use another bidirectional LSTM which takes as input the sentence vectors to get the sentence-level hidden states. The following equations compute the sentence representation for the j^{th} sentence consisting of w words.

$$\begin{aligned} \boldsymbol{sent_j} &= \frac{1}{w} * \sum_{i=1}^{w} h_i \\ \vec{h}_j &= \overrightarrow{LSTM}\left(\boldsymbol{sent_j}, \vec{h}_j^{prev}\right) \\ \overleftarrow{h}_j &= \overleftarrow{LSTM}\left(\boldsymbol{sent_j}, \overleftarrow{h}_j^{prev}\right) \\ h_j &= \left[\vec{h}_j, \overleftarrow{h}_j\right], j \in [1, M] \end{aligned} \tag{3.2}$$

3.2.3. Importance Estimation Mechanism

Not all parts of an article contribute equally to the summary formation, so we use a mechanism to let the selector learn to classify summary-worthy sentences based on several factors of sentence importance which in addition will help the abstractive model generate the correct sequence of tokens. For each sentence, *j*, in an input article, the model predicts the importance of the sentence using the following equations:

$$\begin{aligned} f_j &= W_j\, h_j + W_p p_j + h_j^T W_r\, a - div_j + b_j \\ div_j &= ReLu\left(W_e[sc_j, h_j]\right) \\ sc_j &= \sum_{k=1}^{j-1} h_k \delta_k \\ \delta_j &= \frac{1}{1 + e^{-(f_j)}} \end{aligned} \tag{3.3}$$

Specifically, a logistic layer determines the contribution of each sentence in the input document. Here, the term $W_j\, h_j$ captures the information each sentence carries, while $W_p\, p_j$ adds sentence positional information which has shown to be an indicative factor especially in a news-style corpus [8, 9]. The sentences are also scored based on how relevant they are to the overall article, as denoted by $h_j^T W_r\, a$, where the article representation a is computed by averaging all the sentence-level hidden states and passing the output through a fully connected layer. In addition, the feature, div_j, ensures that the sentences that are scored highly do not cover the same information. To compute div_j, a feed forward layer with a rectified linear unit (ReLu) non-linear activation function is used, with as input the concatenation of h_j, which is the current sentence representation, and sc_j, which keeps track of the information of all the previously classified sentences up to the j^{th} position. During training, the true labels of the previously classified sentences are used and during testing, the predicted probabilities of the model are used. The normalized importance score, δ_j, is then computed for the sentence using a *sigmoid* function which assigns a value between 0 and 1. Our selector is like the RNN-based sequence model used by Nallapati et al. [8] where a logistic function is generally used in an extractive summarization task to classify the sentences as belonging to the extractive summary or not, but we use these sentence probabilities as an additional input to the attention mechanism to ensure that salient parts of source text are weighted more than non-salient parts when generating the abstractive summary. W_j, W_p, W_r, W_e are all weight matrices and b_j is the bias vector, where all these model parameters are updated during training. Also, note that the positional sentence embedding p_j is computed in a similar way as a word embedding using an embedding matrix $E_s \in \mathbb{R}^{m_s \times D_s}$ where D_s represents the embedding dimensionality. Since input articles have a varying number of sentences, we limit our matrix to a predefined number of sentences, denoted by m_s. For our experiments, we set m_s to the maximum number of sentences of 50 based on our dataset statistics in Table *4.1*.

3.3. Pointer Generator Networks Abstractor

While the hierarchical content selector is used to score input sentences using the important estimation mechanism, a separate abstractive model is used to generate the final summary. Although the standard sequence-to-sequence models with attention have shown to be a powerful solution to abstractive summarization [10, 11, 12], the generated summaries sometimes reproduce factual details inaccurately (e.g. *Germany beat Argentina **3-2***) which is especially common for OOV or rare words that appear infrequently during training. To deal with this problem, we use pointer generator networks [59] so that OOV words are copied from source text via a pointing mechanism, while novel words are generated from the fixed vocabulary through the generator. We also use a technique called *coverage* proposed by [59] to ensure that the network doesn't attend to the same input parts when decoding to avoid outputting repetitive summaries (e.g. *Germany beat Germany beat Germany beat…*). We next give details on our abstractive model and how it is combined with the hierarchical selector model.

3.3.1. Encoder

To get the input representation, *I*, a bidirectional LSTM encoder is used which goes through the words of the given document, *x*, to compute their hidden states and the last encoder vector s_N is used as *I*. The k^{th} word will be computed as follows,

$$\begin{aligned} \vec{s}_k &= \overrightarrow{LSTM}\,(\boldsymbol{x_k}, \vec{s}_k^{prev}) \\ \overleftarrow{s}_k &= \overleftarrow{LSTM}\,(\boldsymbol{x_k}, \overleftarrow{s}_k^{prev}) \\ s_k &= [\vec{s}_k, \overleftarrow{s}_k], k \in [1, N] \end{aligned} \tag{3.4}$$

3.3.2. Decoder

After encoding an input into an intermediate representation, *I*, a summary is generated, one target word at a time, using a unidirectional LSTM with one hidden layer. Hence *log P(y|x)* can be decomposed to predict individual word-level probabilities, conditioned on the input representation and what the model has produced so far, as shown below,

$$\log P(y|x) = \sum_{t=1}^{|y|} \log P(y_t|y_{<t}, I)$$

Additionally, we include an attention function to allow the decoder to refer to the source text at each step of prediction. Different from previous attention models, we compute attention using multiple levels of input so that the model selectively focuses on content important to the output generation.

Multi-level attention mechanism

The hierarchical architecture lets our model cope with long textual sequences by calculating the context vector using the following equations:

$$c_t = \sum_k att_k^t \, s_k \tag{3.5}$$

At each decoder time step *t*, the output prediction p_{vocab}^t depends on the output state $\acute{d}_t$ and context vector c_t, which is a sum of the encoder hidden states $(s_1, s_2, s_3, \dots, s_N)$ of the input sequence weighted by alignment scores. The normalized attention score between the output hidden state at t^{th} step and the input hidden state at position *k* is computed as,

$$att_k^t = \frac{e^{u_k^t}}{\sum_{\acute{k}=1}^{N} e^{u_{\acute{k}}^t}} \tag{3.6}$$

where u_k^t is the alignment scoring function computed using the additive form [1], as shown below,

$$u_k^t = V_a^T \, tanh\left(W_t d_t + W_k s_k + W_q q_k + b_a\right) \tag{3.7}$$

Here, *q* is an additional vector containing the sentence-level probability scores of the input words, where the k^{th} word belonging to the j^{th} input sentence is assigned the score $\delta_j(k)$. This allows the model to not only know how well the input words around position *k* match with the given decoder state d_t, but also get access to the input sentences that have a high affinity to the recurrent state at the current time step. Some previous models [10, 16] use a similar hierarchical attention method, but they first compute the word-level attention distribution, then reweighted the word-level attention by the sentence-level attention and renormalized the scores to get the final attention probabilities. In our case, we use the sentence-level information when computing the word-level attention probabilities to better

inform the decoder where to focus on in the input article during summary generation, rather than recomputing the word-level scores in a post-step.

Using the recurrent update function with LSTM units, a new decoder state $\acute{d}_t$ is then generated from d_t and a matrix concatenating the fixed-size input representation c_t to $\boldsymbol{y_t}$, which is the word embedding of previous output:

$$\acute{d}_t = LSTM(d_t, [\boldsymbol{y_t}; c_t]) \tag{3.8}$$

Note that during the testing phase, the previously generated word is fed as input to the next step but for the training phase, $\boldsymbol{y_t}$ is the ground truth word. We set $d_0 = s_N$ and update the learnable parameters V_a, W_t, W_k, W_q, and b_a at each training step.

Generated Summary

To compute the vocabulary distribution, p^t_{vocab}, a *softmax* activation function is used to output the likelihood of each word in the vocabulary being the next word in the sequence,

$$p^t_{vocab} = softmax\big(W_v[\acute{d}_t, c_t] + b_v\big) \tag{3.9}$$

In the pointer generator networks, an additional generation probability p_{gen} is introduced at every decoder step *t* to weight and combine the vocabulary distribution p_{vocab} and attention distribution *att* into a final distribution p_{final} using the following equation,

$$p^t_{final} = p_{gen}p^t_{vocab} + \big(1 - p_{gen}\big)att^t \tag{3.10}$$

By computing a probability distribution over the union of all words in the vocabulary and those appearing in the source document, the model can recover original words by putting sufficiently large attention on the relevant words and making p_{gen} sufficiently large. To calculate p_{gen}, a sigmoid function is used via the following formula,

$$\begin{gathered} p_{gen} = \sigma\big(W_c c_t + W_d \acute{d}_t + W_y \boldsymbol{y_t} + b_{gen}\big) \\ p_{gen} \in [0,1] \end{gathered} \tag{3.11}$$

Weight matrices W_v, W_c, W_d, W_y, and bias vectors b_v, b_{gen} are all learnable parameters.

To eliminate covering same words of input text repeatedly, a coverage mechanism penalizes the network from summarizing to the same parts again by summing up the attention probabilities that each input word has received so far at the current decoder step *t*,

$$cov = \sum_{\acute{t}=1}^{t-1} att^{\acute{t}} \tag{3.12}$$

We add this coverage vector to our attention mechanism by updating *equation 3.7*, hence at the t^{th} decoder step and for the k^{th} encoder word belonging to the j^{th} sentence in the input, the updated equation is:

$$u_k^t = V_a^T \, tanh\left(W_t d_t + W_k s_k + W_q q_k + W_{cov} cov_k + b_a\right) \tag{3.13}$$

where W_{cov} is a learning parameter vector and the coverage at the first time step is a vector of zeros.

Thus, our combined model ensures to attending words from key sentences more than other sentences while discouraging visiting the same source locations repeatedly during summarization.

3.4. Training

Using a composite loss function, the errors made by our system during training is minimized and the model parameters are optimized from a dataset containing *K* article-summary pairs:

$$loss = \frac{1}{K} \sum_{n=1}^{K} loss^n \tag{3.14}$$

$$loss^n = loss_w^n + \theta_s loss_s^n + \theta_c loss_c^n \tag{3.15}$$

Here $loss^n$ is the total loss for the n^{th} pair (x^n, y^n) in the labeled data. Our training objective consists of sub-loss functions reweighted by hyperparameters θ_s and θ_c, that are explained below.

$loss_w^n$: The negative conditional log-likelihood of the generated sequence of words of length $|y^n|$ is computed with respect to the ground truth summary,

$$loss_w^n = -\frac{1}{|y^n|}\sum_{t=1}^{|y^n|} \log p(y_t^n | y_{<t}^n, x^n) \tag{3.16}$$

$loss_s^n$: A cross-entropy loss function for every article with *M* sentences is also computed by comparing the predicted sentence probabilities δ_j with the reference labels g_j,

$$loss_s^n = -\frac{1}{M}\sum_{j=1}^{M} g_j \log \delta_j + \left(1 - g_j\right) \log\left(1 - \delta_j\right) \tag{3.17}$$

$loss_c^n$: Additionally, a coverage loss is used at every training step to penalize any overlap between the coverage and attention vectors for every t^{th} decoder step and across all the *i* encoder steps,

$$\begin{aligned} loss_c^n &= \frac{1}{|y^n|}\sum_{t=1}^{|y^n|}\sum_{i} minimum(att_i^t, cov_i^t) \\ loss_c^n &\leq \sum_i att_i^t = 1 \end{aligned} \tag{3.18}$$

As mentioned earlier, Figure *3.1* demonstrates a block diagram of our end-to-end model trained with the composite loss function. Given a source document, the hierarchical selector and the abstractive models are combined through the multi-level attention function and trained jointly to minimize all the sub-losses.

To verify the usefulness of the end-to-end model, we use another training procedure whereby the hierarchical model and abstractive model are trained separately, as shown in Figure *3.3*. Here, we firstly feed the source document to the content selector and train it

with the selector loss, $loss_s^n$, to identify the top-r most salient sentences. After that, the abstractor is trained with only the top-r sentences as input in a separate pass to get the input representation, instead of feeding the entire source document to the encoder, and word-level loss, $loss_w^n$, with the coverage loss, $loss_c^n$, are then minimized. The decoder uses the standard attention mechanism without utilizing information about sentence importance scores.

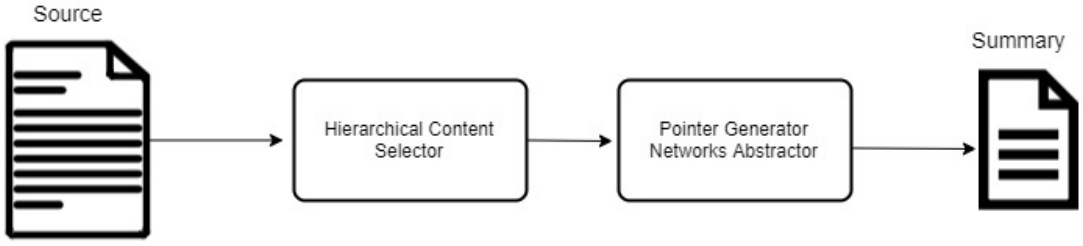

Figure 3.3*:* Block diagram of a hierarchical content selector and pointer generator networks abstractor trained separately

Chapter 4.

Experimental Settings

4.1. Dataset

There are several existing datasets containing newswire article-summary pairs that are often used in summarization tasks, such as the DUC datasets, Gigaword corpus, New York Times corpus [5], NEWSROOM dataset [6], and CNN/Daily Mail combined dataset. DUC datasets have been widely used for different summarization tasks in previous research works, however, the datasets are small containing around 500 documents with on average 35.6-word tokens and summaries with on average 10.4-word tokens. Hence, the dataset size makes it difficult to use them as training data in neural-based models. Gigaword and NEWSROOM are large-scale corpora containing millions of articles from diverse newswire sources. But the summaries in Gigaword are very short consisting of 8.3-word tokens on average, hence more suitable for headline generation task. The NEWSROOM dataset consists of summaries with 26.7-word tokens on average, where most of the summaries are single sentences. While the New York Times corpus contains longer summaries, the articles, published between 1997 to 2007, were collected from a single source and the summaries were written by library specialists instead of being written at the time of publication. Hence, for all our experiments we train and evaluate our models on the CNN/Daily Mail news corpus, which contains a large collection of long newswire articles, published between 2007 to 2015 and curated from two different sources. The human-generated bullet points from articles in CNN and Daily Mail websites are concatenated to obtain summaries expanding multiple sentences. The combined dataset was originally used by Hermann et al. [72] for the task of passage-based question answering and modified by Nallapati et al. [10] to be used for abstractive text summarization. We use the scripts provided by [59] to retrieve the deanonymized dataset with actual entity names. Details of the training, validation, and test splits are provided in *Table 4.1*. We use Stanford CoreNLP tools [70] to preprocess the data, which includes tokenizing the data and applying a lower case to the tokens to reduce the vocabulary size.

For the selector model we use the same dataset and to find the binary ground-truth labels for the articles in the training set, we follow a similar approach proposed by [8]. Since we use this model as an intermediate stage to help the abstractive model in selecting relevant information from the input document before generating the final summary, we compute the ROUGE scores between all the sentences in an article and the corresponding reference abstractive summary. The sentences are then sorted according to their scores in descending order. To identify a subset of key sentences for an article, we use a greedy approach and select sentences one at a time from the sorted list such that it maximizes the total recall value of the ROUGE score for all the selected sentences and continue this process until the overall score can't be improved by the remaining sentences anymore. Hence, for every article in the training set, we get, g, which is a labeled sequential list of 1's and 0's, where the selected sentences are labeled as 1 while the rest of the sentences are labeled as 0.

Table 4.1*:* Statistics of the CNN/Daily Mail dataset. Information includes the number of document-summary pairs, the average number of sentences per document, the average number of sentences per summary, the average number of words per document, and the average number of words per summary in the training, validation, and test sets.

	Size	**Avg input sentences**	**Avg output sentences**	**Avg input words**	**Avg output words**
Training	287226 pairs	31.6	3.8	791.7	55.2
Validation	13367 pairs	26.7	4.1	769.6	61.5
Test	11490 pairs	27.1	3.9	778.6	58.3

4.2. Implementation Details

Our models are implemented using the Tensorflow [14] framework, which is commonly used for text-based applications with sequential information like textual summarization. The hyperparameters of the models are tuned on the validation set.

For the selector, since the dataset contains 30 sentences on average and 36 words per sentence on average, we set the maximum number of sentences to 50, with the maximum number of words per sentence also being set to 50. The hidden state size is set to 200; we experimented with a higher value of 400 hidden units and dropout, but the results didn't change much. For the abstractor, we follow the same configuration settings as [59] used for the pointer generator networks. We use 256-dimensional hidden states for both Bi-LSTM encoder and unidirectional LSTM decoder. We limit the lengths of input and output texts to 400- and 100-word tokens respectively during training while the summary length is increased to 120 tokens during the testing phase.

For both the models, the word vocabulary is constructed using the top 50,000 words in *V*. We use Adagrad [15] as our optimization algorithm during training, with an initial learning rate of 0.15 and a batch-size of 16 where the training data is randomly shuffled at every epoch. Regularization is applied in the form of gradient clipping and no dropout is applied. To initialize the word embedding vectors, we experiment with two settings: 1) the word vectors are initialized from scratch with embedding size of 128, and the values are updated during training, and 2) all the words in *V* are mapped to fixed 200-dimensional pre-trained word embeddings from GloVe [69] which consists of 400,000 unique words trained on the Wikipedia 2014 and Gigaword datasets. From the 50,000 words in our vocabulary, 48,428 words were initialized using the GloVe embeddings, with a vocabulary coverage of 96.86%, while the OOV words were initialized randomly from a normal Gaussian distribution. For both settings, the encoder and decoder sequences share the same embedding matrix.

To give a head start to our end-to-end model, we pretrain the selector and abstractor with the above settings before combining the models, so that the end-to-end model is initialized with the learned weights restored from the Tensorflow checkpoint files of the pretrained

models. The selector is trained for about 40,000 iterations and validated simultaneously and the configuration settings of the best model on the validation set is used to initialize the pretrained selector during end-to-end training. The abstractor is trained without coverage mechanism for about 90,000 iterations and in a similar method to See et al. [59], the input and output sequence lengths (e.g. *max_enc_steps* and *max_dec_steps*) are increased incrementally during training as follows – the abstractor starts with *max_enc_steps*=50 and *max_dec_steps*=15 for faster iterations, and then adds 50 more timesteps to *max_enc_steps* and 10 more timesteps to *max_dec_steps* for every additional 10,000 iterations, until reaching *max_enc_steps*=400 and *max_dec_steps*=100 where the model is trained with the coverage loss. See et al. [59] mentions that training with coverage loss from the beginning reduces the overall performance, hence we separately train the model with coverage loss for the last few iterations after the convergence of the abstractor model. The learned values of the best model on the validation data are used to initialize the parameters of the pretrained abstractor during the end-to-end training.

For the end-to-end model, the selector and abstractor weight parameters are initialized with the learned values from the pretrained models and the hyperparameters in the losses are set as follows: $\theta_s = 5.0$ and $\theta_c = 1.0$; we also experimented with a lower value of θ_s but it didn't help in improvement when combined with the abstractor. Since the articles in the dataset contain 778.6 words on average and the summaries contain 58.3 words on average, we set the lengths of input and output texts to 800- and 100-word tokens respectively. A batch size of 8 is used with a learning rate of 0.01 and the combined model is jointly trained with the overall loss function for 30,000 iterations. The loss is tracked with the validation data simultaneously and the training is stopped when no further decrease in the evaluation loss is observed.

For all our experiments, the models are trained on Nvidia GeForce GTX 1080 GPUs. It took 13 hours to train the selector model with 7,767,001 parameters, 23 hours to train the abstractor model with 21,501,777 parameters, and 1 day and 2 hours to train the end-to-end model with 29,258,378 parameters. While using the standard pointer generator networks took more than 3 days to train for a total of 12.8 epochs, we observe that our end-

to-end model requires a smaller number of epochs to converge since we use learned values from the pretrained submodules to initialize the weights of the end-to-end model.

At test time, in order to generate the most likely output sequence, since exhaustively searching through all possible candidates based on their likelihood is exponential in the summary length, we opt for an approximation algorithm like beam search to generate the *k* most probable output sequences at each decoding time step.

Chapter 5.

Experimental Results

5.1. Evaluation Metric

We measure the performance of our automatic summarization models by reporting both the quantitative and qualitative results on the CNN/Daily Mail dataset. We give an overview of the main quantitative approaches used in summarization evaluation for this research work.

Since our end-to-end model is composed of two intermediate models of selector and abstractor, the pretrained models are firstly evaluated using different quantitative measures like accuracy, precision, and recall, before training the modules together and calculating the performance of the combined model. Classification accuracy is used to find the number of correct predictions made by the selector to choose summary-worthy sentences from each article in the test set using,

$$Accuracy = \frac{\sum True\ positive + \sum True\ negative}{Total\ number\ of\ article\ sentences}$$

True positive and true negative values are calculated using the following chart,

		Predicted	
		Selected	**Not Selected**
Reference	**Selected**	True positive	False negative
	Not Selected	False positive	True negative

Since a summary is a condensed form of an article, there are a smaller number of reference selected sentences than the number of unselected sentences for the training labeled set. Due to this imbalance in the summary membership of the sentences, we introduce a condition

during the testing phase to select at least a minimum number of salient sentences using the following rule,

$$selected_{sents} = sum\big(selected_{probs} \geq threshold\big)$$

$$if\ selected_{sents} < min\_select\_sents\ :$$
$$selected_{sents} = min\_select\_sents$$

Here, the *threshold* is set to 0.5 to include summary-worthy sentences and *min_select_sents* is set to 5 based on the validation set. Hence, if the number of selected sentences is lower than the minimum number of sentences, then sentences with the highest probability are included until exceeding the length limit of 5. We report the overall accuracy by computing the average accuracy over all articles in the test set of CNN/Daily Mail corpus. In addition, we also use more reliable metrics like average precision and recall scores over the test set to evaluate the selector model.

To evaluate our abstractive and end-to-end models, we use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [30] which is an intrinsic evaluation metric to compare automatically generated summaries against a set of human-authored reference summaries. It is an *n*-gram overlap based measure, like BLEU (Bilingual Evaluation Understudy) [31] used for machine translation, but ROUGE is more suitable for summarization and content selection-based tasks since it is recall oriented. Recall and precision are calculated using *n*-gram overlap, where recall computes the fraction of reference summary recovered by the system summary while precision computes the fraction of system summary that is correct,

$$Recall\ = \frac{number\ of\ overlapping\ tokens}{tokens\ in\ reference\ summary}$$

$$Precision\ = \frac{number\ of\ overlapping\ tokens}{tokens\ in\ system\ summary}$$

While the generated summaries may include all tokens in the reference summaries, giving perfect recall, they may also consist of many other words making the summary unnecessarily verbose with low precision. Hence, for all our experiments we report the F-Measure, which takes the harmonic mean of the recall and precision. ROUGE has different

parameter settings, including word stemming, stop word removal, and *n*-gram size; we report ROUGE-1 and ROUGE-2 scores which consider unigram and bigram-overlaps between the system and gold-standard summaries, as well as ROUGE-L score which measures the longest matching sequences of tokens using longest common subsequence (LCS).

We also evaluate the generated abstractive outputs using the METEOR (Metric for Evaluation of Translation with Explicit Ordering) automatic evaluation metric of [4], which calculates both segment and system-level scores based on alignments between hypothesis-reference pairs, but unlike ROUGE, the metric considers synonym and paraphrase matches between words and phrases along with exact matches. Here, the final score is based on a weighted harmonic mean of unigram precision and recall as well as a penalty function for incorrect word order,

$$Final\ Score = FMean\ * (1 - Penalty) \tag{5.1}$$

FMean, which only considers unigram matches, and the fragmentation penalty, which takes longer *n*-gram matches, are calculated using the following equations:

$$FMean = \frac{Precision * Recall}{(\alpha * Precision) + ((1 - \alpha) * Recall)} \tag{5.2}$$

$$Penalty = \gamma * \left(\frac{c}{m}\right)^{\beta}, \text{where } 0 \leq \gamma \leq 1 \tag{5.3}$$

Here, *c* is the fewest possible number of chunks, where a chunk refers to adjacent matches in the same word order, and *m* is the total number of matched unigrams. In addition, α, γ, and β are tuned parameters of METEOR, where α controls the relative weight of precision and recall when calculating the FMean score, while γ and β control the weight and shape of the fragmentation penalty respectively. With the penalty function, if most of the matches are contiguous, then the number of chunks is lower, and the penalty decreases which leads to an increased final score.

5.2. Results and Discussion

5.2.1. Quantitative Results

To determine how each sentence feature contributes to the overall performance of the selector model, *Figure 5.1* shows the average accuracy, precision, and recall scores over all the articles on the test set of CNN/Daily Mail corpus using a different combination of features. Our results reveal that adding information about how salient a sentence is with respect to the input document helps the model select more relevant sentences compared to when only using information about the sentence representation. This can be seen by the increase in recall value from 44.0 to 53.9. We also observe that while including positional features improves performance in terms of accuracy and precision, there is a slight decrease in recall. On analyzing, we see that on average more sentences are selected by model with content and relevance features than the model with added positional features. Out of all the ground-truth selected sentences, 53.9% were correctly classified by the former model while 52.7% were correctly classified by the latter model, but the former model also selected more unnecessary sentences. Including the positional information helps the model to make a good balance between precision and recall. It can also be seen that including the previous summary information to inform the next sentence predictions increases the recall score to 52.9, while the precision and accuracy scores are 64.4 and 75 respectively.

In *Figure 5.2*, we analyze the sentence position distribution of the sentences selected by our selector model during the testing phase. We see that about 33% of the sentences selected by our model belong to the first 3 sentences and around 21% of the sentences are from positions 4, 5, and 6. These statistics are in accordance with previous studies on sentence extraction [8], which states that newswire articles tend to be organized with the main information appearing within the first few sentences of the starting paragraph. From the figure, we can also see that the model picks a decent number of sentences from the sentence positions in the range of 7 to 12, making up about 20% of the total retrieved sentences. Each of the remaining sentence positions covers less than 2% of the selected sentences and very few sentences are selected from the last positions since the average length of the articles is 30 sentences.

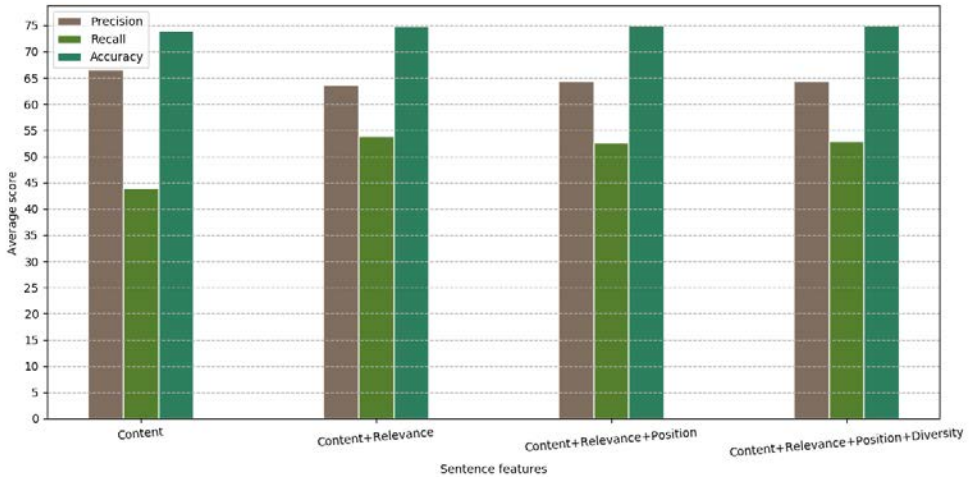

Figure 5.1*:* Bar chart comparing the average scores of accuracy, precision, and recall for the sentence prediction model with a different combination of features on the test set of CNN/Daily Mail corpus

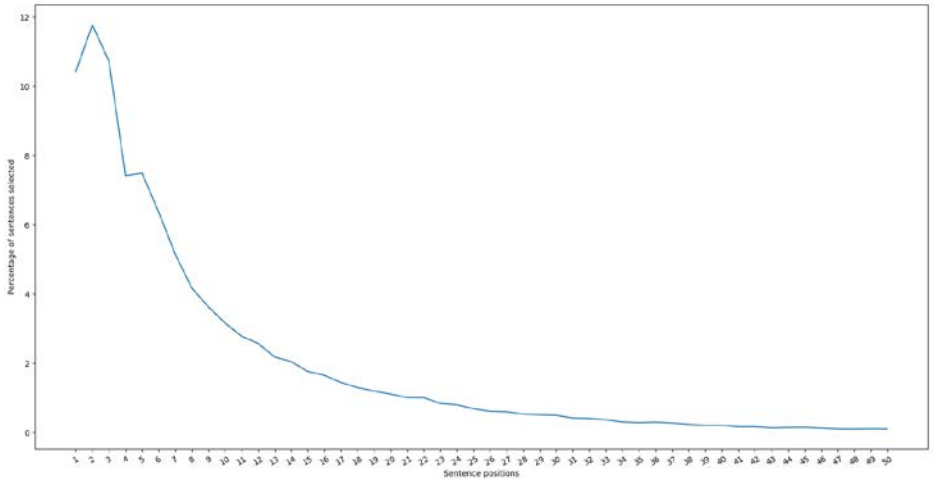

Figure 5.2: Plotting the percentage of sentences selected during test time against the sentence positions. We set the maximum number of sentences for all documents to 50.

Our end-to-end models are evaluated on the test set of CNN/Daily Mail corpus, consisting of 11,490 document-summary examples not seen by the models during the learning phase. To check which beam-size value outputs the most likely sequence with the beam search algorithm, we vary the number of candidates to consider at each decoder step. *Table 5.1* shows the ROUGE-N (N=1, 2) and ROUGE-L F1 scores for different beam-width values on the test dataset. Different from previous abstractive models [16, 59, 63] which use a

beam size of 4 and 5 to report their best scores, we can see from the table that the larger beam-width of 7 results in a better performance of our end-to-end model as the multiple candidate sequences increase the likelihood of better matching a target sequence. However, increasing the beam width parameter further doesn't produce more reliable scores and tends to generate degraded output, as depicted by the example in Figure *5.10*.

Table 5.1*:* ROUGE scores of the end-to-end model with varying beam-width values

Beam Width, *k*	ROUGE-1	ROUGE-2	ROUGE-L
k = 4	40.40	17.78	36.92
k = 5	40.40	17.79	36.93
k = 6	40.42	**17.84**	36.94
k = 7	**40.44**	17.82	**36.95**
k = 8	40.35	17.74	36.88
k = 9	40.38	17.74	36.89
k = 10	40.41	17.77	36.91

We also trained our model with different selector loss weights, θ_s, while keeping θ_c to 1 following the same configuration setting as in [59]. We can see from *Table 5.2* that weighing the sub-losses equally with $\theta = 1$ and using lower values of θ_s didn't change the overall performance much. Increasing the selector loss hyperparameter to a much higher value like 5 allows the combined model to penalize the losses on mistakes during the training phase, reducing overall loss from an initial value of about 3.8 to about 2.6, to give a better performance.

Table 5.2*:* ROUGE scores of the end-to-end model with varying selector loss weight values, keeping θ_c to 1 and beam size to 7

selector loss weight	ROUGE-1	ROUGE-2	ROUGE-L
$\theta_s = 1$	39.90	17.34	36.40
$\theta_s = 2$	39.78	17.34	36.36
$\theta_s = 5$	40.44	17.82	36.95

We compare three of our models with previous summarization models on the same dataset using the following parameter settings:

1. C1: The selector and abstractor are trained separately with learned embedding vectors.

2. C2: End-to-end model where the pretrained word embeddings are held fixed.
3. C3: End-to-end model with trainable word vectors initialized from scratch.

All other parameter values are the same as mentioned in Section 4.2 of Chapter 4 for the three models; selector loss weight is kept at 5 and output summaries are generated using a beam size of 7.

Table 5.3 shows the existing models used for comparison, details of which are provided below:

Baselines:

1. B1: Lead-3 baseline which selects the first 3 input sentences as summary. This is a strong baseline when dealing with newswire stories since news articles tend to be structured with the most important information at the start. Although this baseline is based on a simple heuristic, it outperforms most of the previous abstractive models.
2. B2: Standard RNN-based encoder-decoder attentional model using 50,000 words in the vocabulary.
3. B3: Standard RNN-based encoder-decoder attentional model using 150,000 words in the vocabulary.

Extractive models:

1. E1 [8]: A neural extractive model to classify input sentences to be included in the summary using sentence-level features.
2. E2 [8]: A neural network model like E1, but the extractor is trained in an abstractive way using only the reference abstractive summaries, without any sentence-level extractive labels.

Abstractive models:

1. A1 [10]: An abstractive model which uses the sentence-level attention scores to override the word-level attention scores dynamically at each step of word generation.
2. A2 [10]: An encoder-decoder RNN model which uses a temporal attention mechanism where the attention distribution is divided by the sum of the previous attentions to reduce repeated phrases in the output summary.
3. A3 [59]: An abstractive model that uses pointer generator networks with coverage mechanisms.
4. A4 [63]: The model uses a switch function to allow copying of words, like A3, but uses a reinforcement-based learning objective during training.

5. A5 [63]: The model is the same as A4 but uses a hybrid learning objective during training.
6. A6 [60]: The model identifies keywords from the input text and then encodes the keywords to a key information representation to guide the process of summary generation.
7. A7 [16]: A unified model where the word-level attention scores are rescaled by fixed sentence-level attention scores to generate the final abstractive summary.

As can be seen in *Table 5.3*, C1 which trains the submodules separately performs better than the baselines, B2 and B3, and the extractive models, E1 and E2. We can see that B3, which uses a larger vocabulary size of 150,000 performs poorly than our model which uses 50,000 vocabulary words. This shows the effectiveness of using pointer generator networks to handle OOV words. Also, A2 which uses a temporal attention function to avoid repeated phrases in the output summary scores less than our model which uses the coverage mechanism to reduce repetitive sequences. While C1 produces results comparable to those by the abstractive models with respect to ROUGE, it did not surpass the score of the lead-3 baseline. But our end-to-end model C3, which trains the hierarchical selector and pointer generator-based abstractor jointly, exceeds the ROUGE F1 scores of the lead-3 baseline and all the extractive models. Hence this empirically shows that our model can identify salient parts of an article with the help of the combined scoring mechanism which lets the generated summaries have a higher rate of *n*-gram overlap with the reference summaries. Compared to C1, using the end-to-end model increases ROUGE-1 by 1.02%, ROUGE-2 by 1.71%, and ROUGE-L by 1.18%, which reemphasizes the need to jointly train the models. Our model C2, which uses the fixed pretrained GloVe word embeddings, scores less than C3, which uses learned word vectors. One reason might be since the end-to-end model initializes the weight vectors with the learned values from the pretrained sub-models, hence making the embeddings trainable allows better word representations to be learned and give better performance.

Next, we compare our best model C3 with other models using hierarchical attention techniques. We can see that C3 performs much better than A6, which uses keywords from input to better inform the attention mechanism. One reason for the low scores might be since A6 trains the keywords extractor model first and then the guided network in a

sequential manner, whereas C3 is trained jointly allowing for better learning. Our results also suggest that our end-to-end model performs better when the sentence-level saliency information is included during the computation of the word-level attention scores rather than recomputing the scores in a later stage. This can be seen by the increase in ROUGE across all scores by our model over A1 and A7. Also, this early intervention helps reduce the training time for the end-to-end model, C3 took about 30,000 iterations while A7 took about 50,000 iterations to converge.

We now compare C3 with other models using pointer networks. We can see that when the hierarchical selector model is added to the pointer generator networks, our combined model outperforms the standard pointer generator networks of A3 by a margin of 0.91 metric points in ROUGE-1, 0.54 points in ROUGE-2, and 0.57 points in ROUGE-L respectively. A5, which uses a mixed learning objective function that combines maximum likelihood training objective with reinforcement-based learning, scores less than C3 across all scores. But A4, which only uses reinforcement-based learning outperforms C3 by a large margin in ROUGE-1 and ROUGE-L scores respectively. This increase in quantitative value should also result in qualitative improvement in the generated summaries, but A4 suffers from readability issues as reported by Paulus et al. [63]. In Figure *5.11*, we can see that although the score of A5 is less than that of A4, the generated summary in the former model is more readable than that of the latter model, which tends to produce short and truncated sentences towards the end of sequences. Hence, reinforcement-based learning models can help increase quantitative scores by optimizing a discrete evaluation metric like ROUGE with ROUGE-L, but they don't guarantee an improvement in the quality and readability of the generated output.

Particularly we find that our end-to-end model ensures keeping a good balance between content selection and language fluency– while the selector module helps improve recall (e.g. recall of 47.26 in ROUGE-1, 20.75 in ROUGE-2, and 43.15 in ROUGE-L) so that all the important information that needs to be summarized is indeed summarized, the abstractive module with the word-level learning supervision ensures good language flow (e.g. F1 of 17.82 in ROUGE-2 and 36.95 in ROUGE-L), making the overall summary more coherent and readable.

Table 5.3*:* ROUGE scores evaluated on the test set of CNN/Daily Mail corpus

Models	ROUGE-1	ROUGE-2	ROUGE-L
Baseline			
B1	40.34	17.70	36.57
B2	31.33	11.81	28.83
B3	30.49	11.17	28.08
Extractive			
E1	37.5	14.5	33.4
E2	39.6	16.2	35.3
Abstractive			
A1	32.75	12.21	29.01
A2	35.46	13.30	32.65
A3	39.53	17.28	36.38
A4	**41.16**	15.75	**39.08**
A5	39.87	15.82	36.90
A6	38.95	17.12	35.68
A7	40.19	17.67	36.68
Ours			
C1	40.03	17.52	36.52
C2	40.03	17.50	36.64
C3	40.44	**17.82**	36.95

Table 5.4 compares the METEOR scores of our end-to-end model C3 to the scores attained from A3 which doesn't use an intermediate selector module. We report results in exact match mode, which rewards only exact matches between system-reference words, as well as in full match mode, which also scores matching stems, paraphrases, and synonyms. Since the parameters of METEOR are tuned for a beam size of 40, the beam size is set to this default value to search for the highest-scoring alignment for all the models. We can see from the table that for both the models, scores are better in full match mode than considering only exact matches. While A3 achieves a final score of 16.73 in exact and 19.04 in full match mode, using an intermediate selector model with the pointer generator networks increases the final scores by 1.86 and 2.04 points for exact and full matches respectively. The empirical results also show that our end-to-end model performs better at identifying the most important content from the source inputs to include in the output summary. This can be validated by the significant gain in the recall values, 13.9% when

taking only exact matches and 13.6% for all word matches. Since the fragmentation penalty went down by about 0.26% for both the match modes, based on the empirical results we can say that our model is able to generate summaries more consistent with reference summaries compared to those produced by A3.

Figure 5.3 displays the segment-level METEOR statistics in full match mode for all the 11,490 articles from the test set for C3 (e.g. System-1) and A3 (e.g. System-2). From the first chart, we can see that more than half of the total articles, that is around 7000 of the summaries from System-2, are assigned final scores in the range from 0.0 to 0.2, while a larger number of the summaries from our System-1 are scored in a higher range from 0.2 to 0.5. As shown in the second chart, for both the models more than 10000 of the generated summaries were penalized heavily in the range from 0.5 to 0.6 for not having longer *n*-gram matches with the reference summaries. We believe one reason for this might be due to the subjective nature of the written summaries. The METEOR metric rewards longer lexical overlaps but doesn't take into consideration the fact that multiple summaries of an input phrased differently can express the same meaning without having any lexical overlap. Hence, a system-generated summary can be good but still be penalized if the wording in the system-generated summary is different from that of the reference summary. The full matching mode handles this problem to some extent by considering synonyms and paraphrases, but the look-up tables are fixed. The third and fourth charts show the precision and recall scores for the summaries– while our summaries on average score in lower range than System-2 summaries in terms of precision, the recall values for most of our summaries fall under a much higher range thus resulting in a higher overall system score.

Figure *5.4* compares the METEOR final scores by output summary length between our end-to-end model C3 (e.g. System-1) and standard pointer generator networks model A3 (e.g. System-2). We observe that our model performs much better for longer summaries expanding multiple sentences with about 50 words on average.

Table 5.4*:* Comparing METEOR scores (in full and exact mode) of our end-to-end model C3 with the scores of A3 which uses the standard pointer generator networks. All scores are evaluated on the test set of CNN/Daily Mail corpus

	Precision	Recall	F1	FMean	Fragmentation penalty	Final Score
A3 (match type = exact)	36.72	36.81	36.76	36.79	54.52	16.73
A3 (match type = full)	41.83	41.83	41.83	41.83	54.49	19.04
C3 (match type = exact)	35.11	41.93	38.22	40.75	54.38	18.59
C3 (match type = full)	39.89	47.50	43.37	46.18	54.35	**21.08**

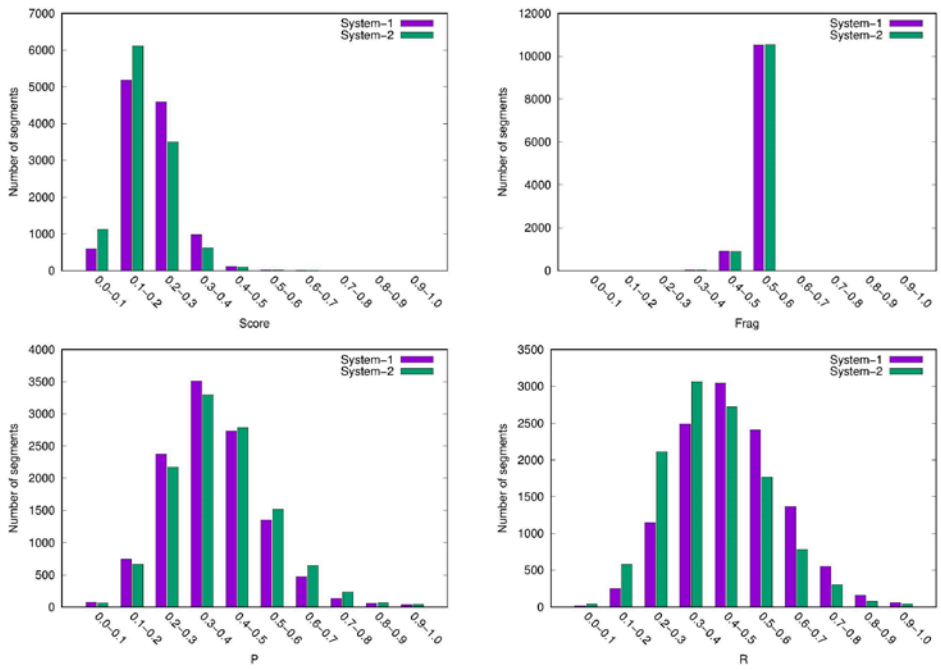

Figure 5.3: METEOR statistics of all the 11,490 generated summaries for our end-to-end model C3 (e.g. System-1) and the standard pointer generator networks model A3 (System-2). The y-axis denotes the number of summaries while the x-axis denotes the scores. The first chart (top left) shows the final scores, the second (top right) shows the fragmentation penalty scores, the third (bottom left) shows the precision scores, and the fourth (bottom right) shows the recall scores. Note that all the scores are shown in decimal points and not converted to a percentage.

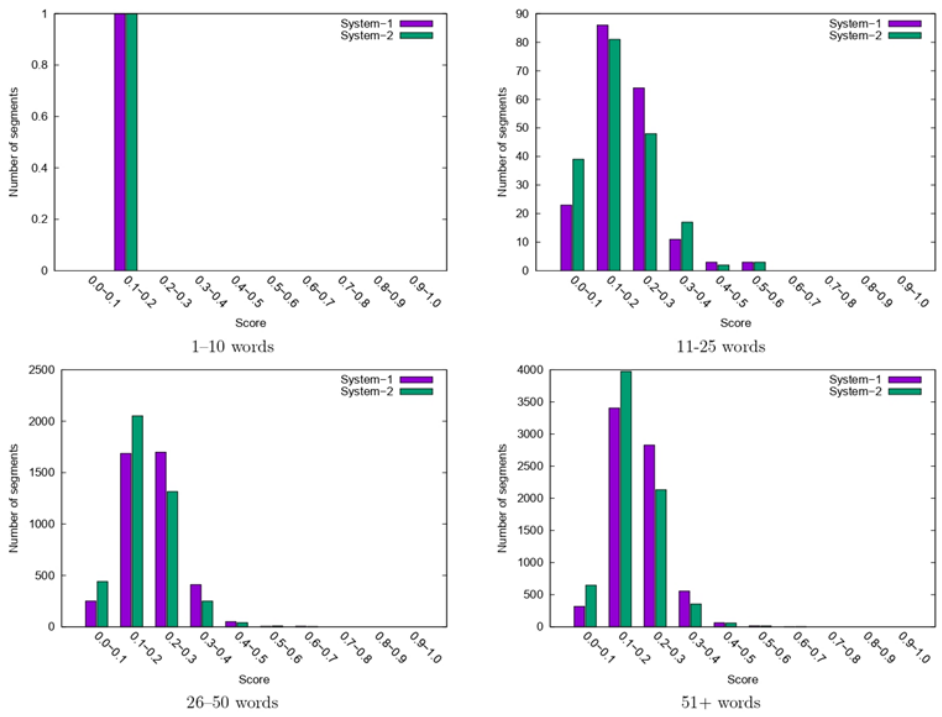

Figure 5.4: METEOR final scores by the summary length of all the 11,490 generated summaries for our end-to-end model C3 (e.g. System-1) and pointer generator networks model A3 (System-2). Note that all the scores are shown in decimal points and not converted to a percentage.

5.2.2. Qualitative Results

Figures 5.5, 5.6 and 5.7 display qualitative results for the segment-level METEOR scores. Figure *5.5* maps the alignment between our generated summary (e.g. left column) and the reference summary (e.g. top row) for a test example where our model scored highly due to longer adjacent mappings. For some examples, our summaries conveyed the same meaning as the corresponding reference summaries but were scored less due to lower lexical overlaps, as shown in Figure *5.7* and Figure *5.7*. The figures compare the alignment scores between models C3 and A3. The top row contains the reference summary while the left and right columns show the summaries generated by C3 and A3 correspondingly. We can see that for both the examples, our model summaries have better alignments with the reference summaries than A3. The summaries generated by A3 consist of mainly secondary and unnecessary information, but our summaries cover only the most important input points resulting in higher precision and recall values. For our model, although some indirect

matchings were rewarded (e.g. yellow boxes) along with the exact matches (e.g. grey boxes), the fragmentation penalty is still in a higher range from 0.4 to 0.6 since there aren't many contiguous sequences and this reduces the final score by about 50%.

P:	0.964
R:	0.964
Frag:	0.326
Score:	0.649

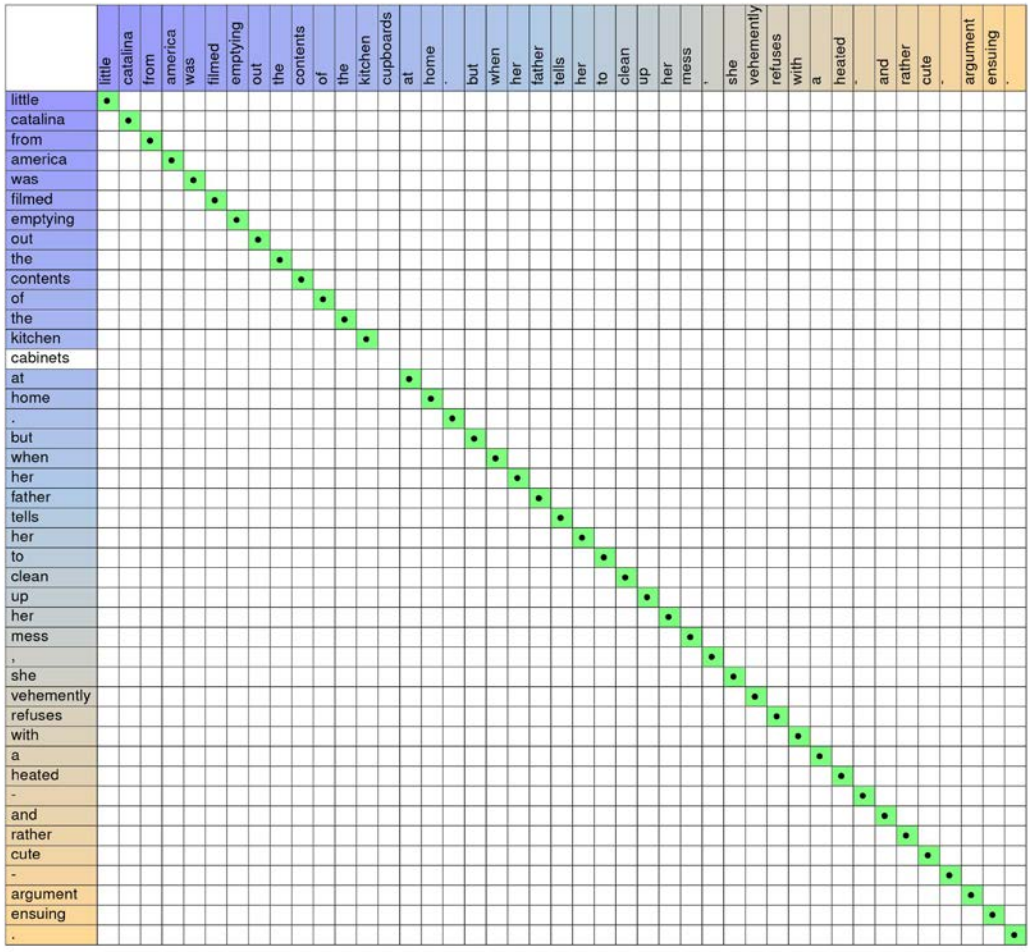

Figure 5.5: Qualitative results of METEOR where our generated summary has high alignment with the reference summary. The top row contains the reference summary while the left column contains the generated summary from our end-to-end model C3. The color spectrum follows reference word order.

P:	0.609	vs	0.285	:	**-0.324**
R:	0.884	vs	0.488	:	**-0.396**
Frag:	0.488	vs	0.539	:	**0.051**
Score:	0.424	vs	0.203	:	**-0.221**

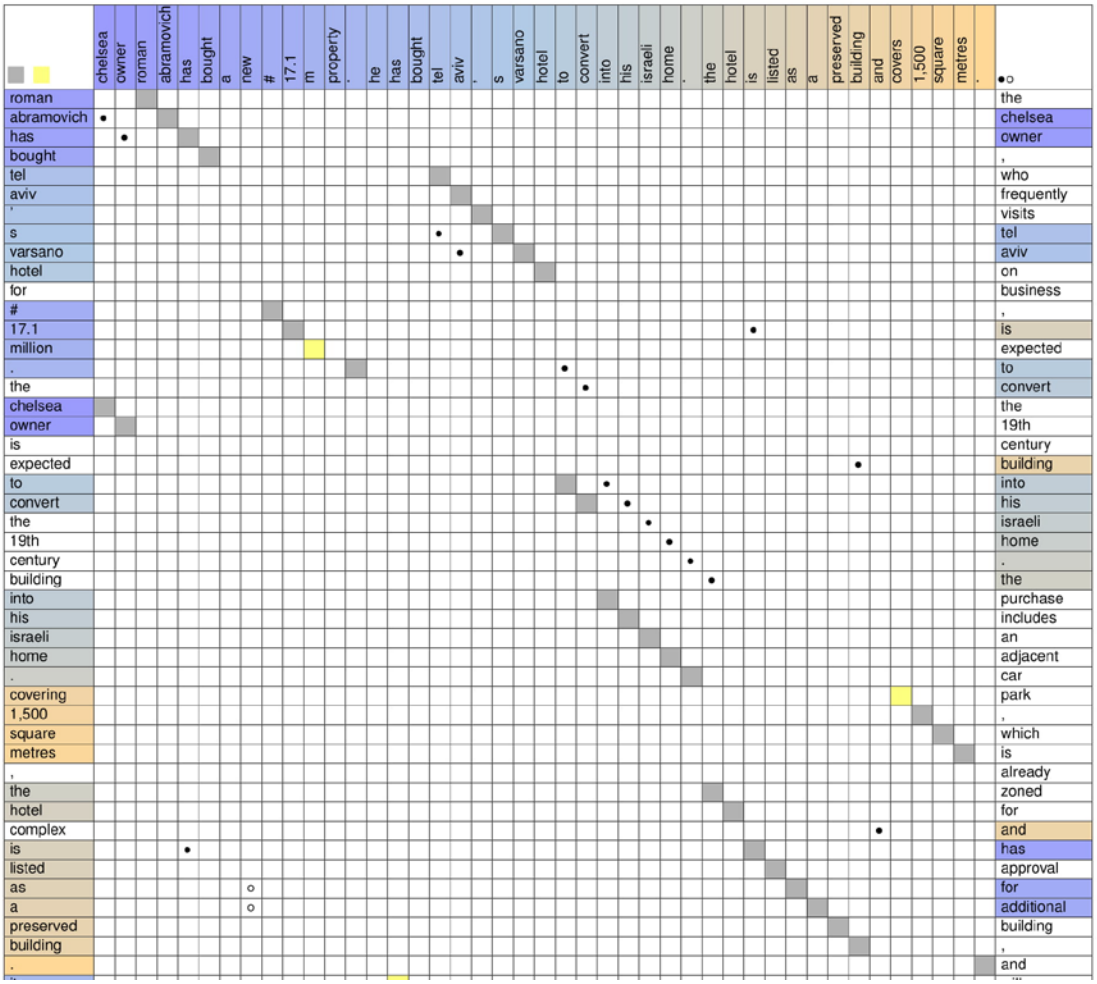

Figure 5.6: Qualitative results comparing METEOR alignment scores between generated summaries from C3 and A3 with the reference summary of example from CNN/Daily Mail test dataset. The top row is the reference summary, left column is our generated summary, and the right column is the generated summary from A3. The color intensity is according to reference word order. The box symbols refer to matches made by our model – exact matches are denoted by grey and full matches are denoted by yellow boxes. The round symbols refer to matches made by A3 – exact matches are shown by black symbols and full matches are represented by white symbols. For the scores, the first column contains our model's METEOR statistics while the other column shows the scores from A3. The scores in red are the difference between the two models.

P:	0.582	vs	0.191	:	**-0.391**
R:	0.833	vs	0.211	:	**-0.622**
Frag:	0.556	vs	0.579	:	**0.022**
Score:	0.347	vs	0.088	:	**-0.260**

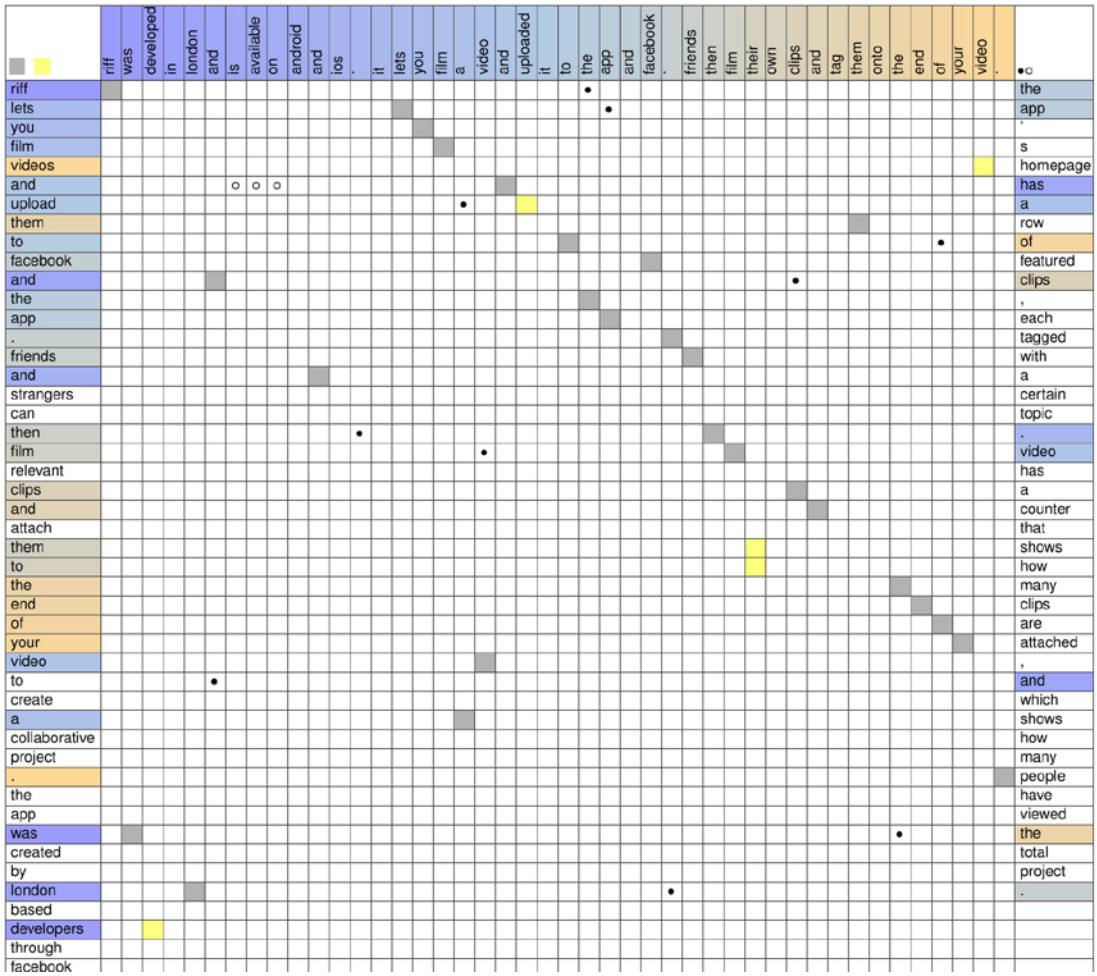

Figure 5.7: Qualitative results for example from CNN/Daily Mail test set where our model is penalized heavily due to not having longer adjacent matches with the reference summary.

To demonstrate the effectiveness of using a novel attention mechanism that utilizes the sentence scores from the selector to modulate the probabilities of the input words being attended to at every timestep of the abstractor decoder, we visualize the attention layer for generated outputs of the test set examples. Figures 5.8 and 5.9 provide two examples from the test set of CNN/Daily Mail corpus. For each example, the reference and generated summaries are provided, and the total attention probabilities assigned by our end-to-end model C3 are also shown. In the attention figure, the rows contain the input sentences and

the columns contain the words per sentence in a document. For visualization purposes, we show only the first 11 sentences and the first 20 words per sentence from each document. The green color denotes the probabilities assigned to the sentences in the selector and the blue color denotes the attention probabilities of the words across all decoder steps of summary generation. The color intensity represents how much weight is given to the words and sentences by our model, hence important sentences and words are darker while the less important ones are lighter in color.

In Figure 5.8, we can see that the human-generated and system-generated summaries convey the same information from the source document. The document consists of 25 sentences (we show only the first 10) and the selector puts the most weight on the first two sentences and the tenth sentence (e.g. *s0*, *s1*, and *s9*) since they hold the main information of the document. Since the sentences are scored based on a combination of features including the sentence position, content, relevance, and diversity, our model can identify key information from anywhere in the document. The combined features also allow the selector to give low probabilities to the longer sentences holding a lot of information but not relevant to the input. The abstractor, on the other hand, uses the hierarchical attention to let the decoder focus on words from the most important sentences, thus improving the overall quality of the summary. We can observe that words from the first, second and tenth sentences are attended most during summary generation and any unnecessary information is ignored by the system. For example, in the second sentence, the first few words (e.g. *the exact time of the cuba trip wasn't immediately released. but the*) are left out, and the decoder only attends to the most descriptive words in that sentence to be expressed in the output text.

In Figure *5.9*, the first two sentences, which only describe the scenario, are given very low probabilities and the selector can classify key sentences (e.g. *s2*, *s3*, and *s8*) from the ones containing no significant information. The abstractor only visits the words in the key sentences to produce the resulting summary, ensuring consistency with the selector. We can see from the decoded output that the constituent, *a city in northern nigeria*, after the proper noun, *zaria*, is dropped by the system to produce a compressed output.

Reference:

trip will come before pope francis arrives in united states .

francis played key role in re-establishing diplomatic ties between cuba and u.s.

Decoded:

pope francis will visit cuba on his way to the united states in september .

the vatican said the pope would stop in cuba before his planned late september stops in washington , new york and philadelphia .

francis , the first pope from latin america , played a role in restarting diplomatic relations between cuba and the united states .

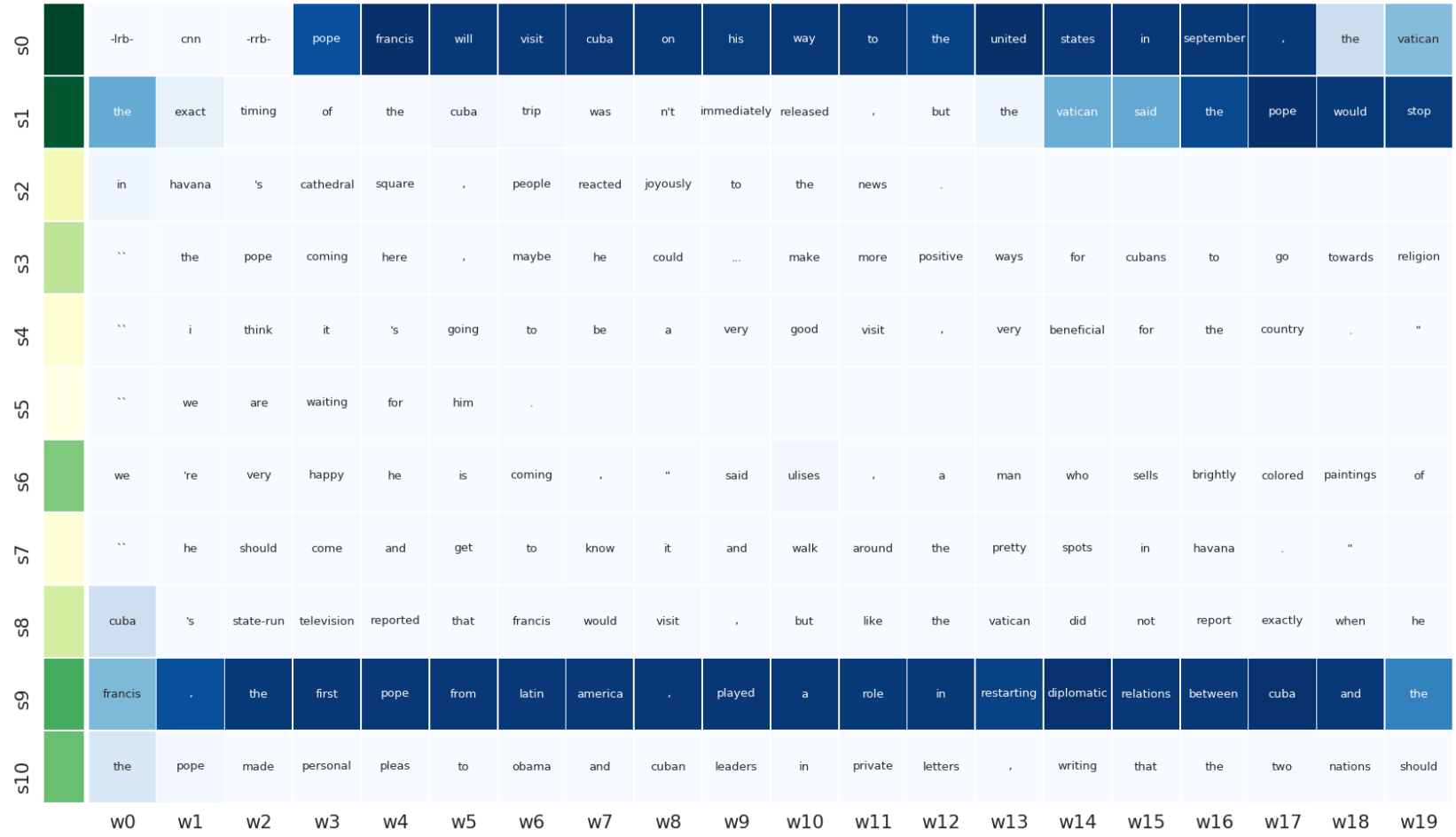

Figure 5.8: ‘Reference’ is the ground-truth summary while ‘Decoded’ is the summary generated using our end2end model C3. For the visualization, color intensity represents weights given to the sentences and corresponding words in the input article. Most important information gets highly scored by our model. Note that the attention score assigned to every word here is the normalized sum over all decoder timesteps. We show the first 11 sentences and the first 20 words per sentence from the document.

Reference:

african students and car enthusiasts are creating eco-friendly cars .

nigerian students will compete in an `` eco-marathon '' in may .

Decoded:

powered by electricity and engineered for efficiency , car enthusiasts from across africa are sparking home-grown concepts that have gotten experts revving .

in zaria , a team of students from the ahmadu bello university are currently applying the final touches to the `` abucar 2 .

'' their compact creation , which was built over five months , will travel to the netherlands in may to compete in the european leg of this year 's shell eco-marathon .

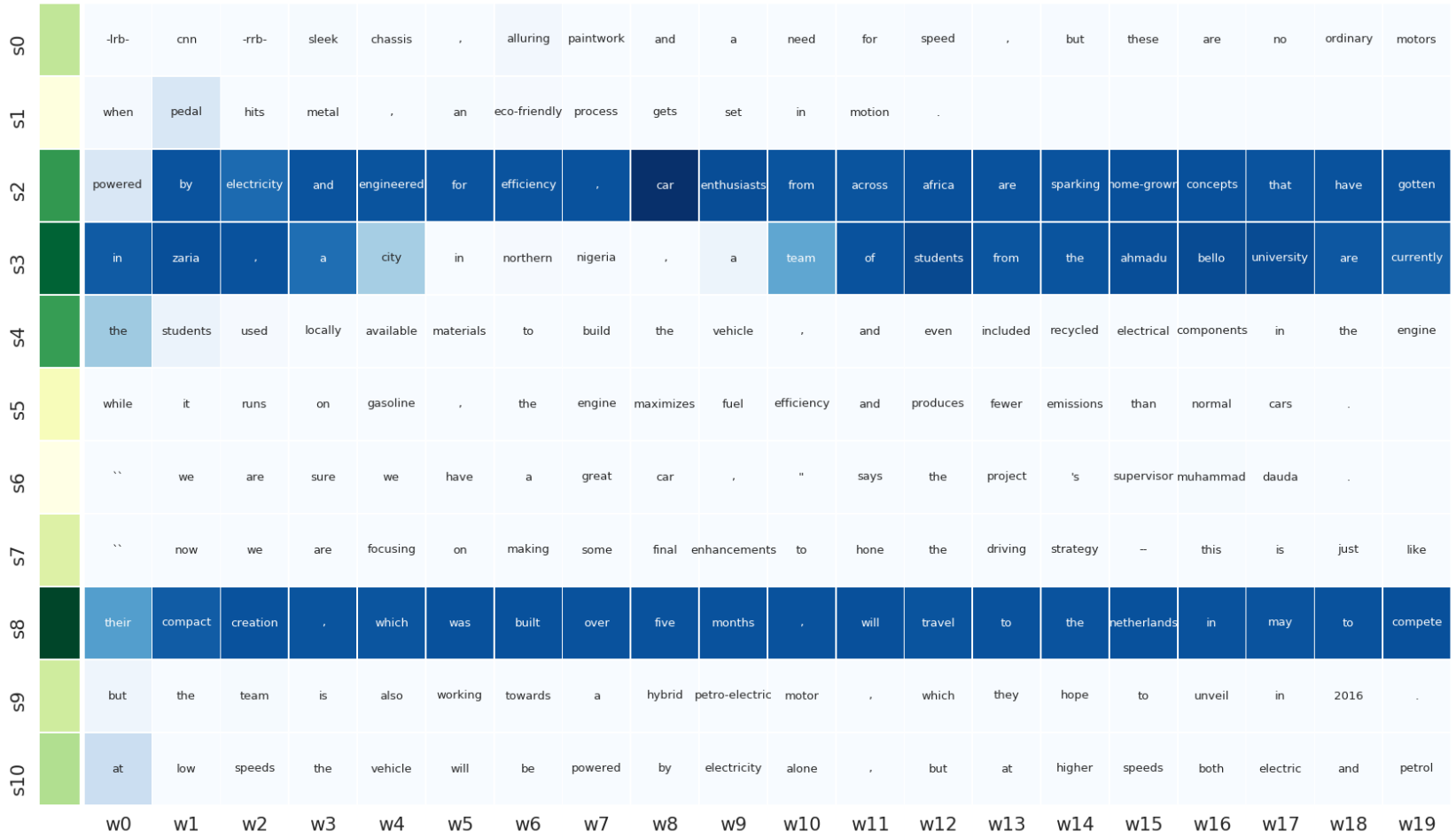

Figure 5.9: ‘Reference’ is the ground-truth summary while ‘Decoded’ is the summary generated using our end2end model C3. In the visualization of the hierarchical attention, rows represent the sentences and columns represent the words per sentence for test set example in CNN/Daily Mail corpus. Only the first 11 sentences and the first 20 words per sentence from the document are shown.

To further validate the qualitative improvement of the resulting summaries of C3 on test data, we compare generated outputs from our end-to-end model C3 and those from models A3 and A7 with the reference summaries. Outputs for A3 was generated using our data split so that the generated test-set outputs can be compared qualitatively. A7 uses the same data split as ours so we use their published results. We couldn't compare our qualitative results with the remaining abstractive models since most of them didn't share their outputs and it was difficult to reproduce their results without enough information for the experimental settings and dataset preprocessing.

Figures 5.12, 5.13, and 5.14 show that our summaries contain more information in agreement with the reference summaries compared to models without the hierarchical architecture. A3 can deal with the OOV problem and use additional coverage to deal with redundant information in the output but lacks in its ability to identify essential information from the source to include in the summary. For example, in Figure *5.12*, the A3 model chooses to include a secondary detail like `` *free the five '' became a twitter hashtag* . rather than including the key point at the beginning of the document, which is the release of the 5 female activists detained last month. Moreover, by inspecting outputs from Figure *5.14* and Figure *5.14*, it's evident that A3 is mostly including words from irrelevant portions of the article. For instance in Figure *5.14*, the model's summary contains no information about the first robot breaking down and the last sentence (e.g. *an indicator that cooling systems were working effectively , according to a statement* .) is unnecessary. The unified model A7, which uses an intermediate extractor module like our C3, can select what content is important from input, but we observe that the generated sentences tend to be longer, sometimes containing additional constituents that do not add any valuable information. Hence a generated sentence will contain the key information plus some additional information, as can be seen from the figures. In contrast, summaries from our end-to-end model mostly capture all the necessary information from fragments of the key input sentences, dropping any additional information from constituent parts. The quality of the summaries from both models is comparable, and in some cases, our summaries are more concise and relevant.

While figures 5.12, 5.13, and 5.14 show summaries of test examples where our model performs well on content selection and can rewrite the selected content in an abstractive way, for some examples we observe that our end-to-end model directly copies the selected input fragments and in some cases whole sentences to include them word-by-word in the summary. Figure *5.15* provides examples of such instances. The highlighted portions in red are the selected input fragments and sentences which are used by the decoder to get the final system summary. In the examples, our end-to-end model can select the right content from the input documents and generate an informative summary but is much less abstractive compared to the human-written reference summaries, which include a lot more rephrasing and generalization.

Input document (truncated): cnn mercedes driver and f1 championship leader lewis hamilton stole pole position for sunday 's chinese grand prix from teammate and fierce rival nico rosberg in dramatic fashion . hamilton took first place on the front row on the last lap , beating rosberg by a slim four hundredths of a second margin . frenemies the two former friends have enjoyed , or rather endured , a heated rivalry since falling out last season and rosberg 's annoyance at hamilton 's last ditch success was obvious . (…)

Reference summary:

lewis hamilton will start on pole in china .

pushed teammate rosberg into second on the last lap .

rosberg refused to shake hamilton 's hand afterward .

Our end-to-end model (beam size=7):

mercedes driver lewis hamilton stole pole position for sunday 's chinese grand prix .

hamilton took first place on the last lap , beating rosberg by a slim four hundredths of a second margin .

german appeared upset as he left his car and refused to shake hamilton 's hand .

Our end-to-end model (beam size=9):

mercedes driver lewis hamilton stole pole position for sunday 's chinese grand prix .

hamilton took first place on the last lap , beating rosberg by a slim four hundredths of a second margin .

kimi raikkonen will join his teammate vettel on the second row and will be looking for a repeat of ferrari 's performance in malaysia .

Figure 5.10: Comparing the quality of the generated summaries of the end-to-end model with varying beam-width values. The underlined part represents a piece of subordinate information.

Source document (truncated): Jenson Button was denied his 100th race for McLaren after an ERS prevented him from making it to the startline. It capped a miserable weekend for the Briton; his time in Bahrain plagued by reliability issues. Button spent much of the race on Twitter delivering his verdict as the action unfolded. 'Kimi is the man to watch,' and 'loving the sparks', were among his pearls of wisdom, but the tweet which courted the most attention was a rather mischievous one: 'Ooh is Lewis backing his team mate into Vettel?' he quizzed after Rosberg accused Hamilton of pulling off such a manoeuvre in China. Jenson Button waves to the crowd ahead of the Bahrain Grand Prix which he failed to start Perhaps a career in the media beckons... Lewis Hamilton has out-qualified and finished ahead of Nico Rosberg (...)

Reference: Button denied 100th race start for McLaren after ERS failure. Button then spent much of the Bahrain Grand Prix on Twitter delivering his verdict on the action as it unfolded. Lewis Hamilton has out-qualified and finished ahead of Mercedes team-mate Nico Rosberg at every race this season. Bernie Ecclestone confirms F1 will make its bow in Azerbaijan next season.

ML (ROUGE-1 41.58): Button was denied his 100th race for McLaren. ERS prevented him from making it to the start-line. The Briton. He quizzed after Nico Rosberg accused Lewis Hamilton of pulling off such a manoeuvre in China. Button has been in Azerbaijan for the first time since 2013.

RL (ROUGE-1 50.00): Button was denied his 100th race for McLaren after an ERS prevented him from making it to the start-line. It capped a miserable weekend for the Briton. Button has out-qualified. Finished ahead of Nico Rosberg at Bahrain. Lewis Hamilton has. In 11 races. . The race. To lead 2,000 laps. . In. . . And. .

ML+RL (ROUGE-1 44.00): Button was denied his 100th race for McLaren. The ERS prevented him from making it to the start-line. Button was his team mate in the 11 races in Bahrain. He quizzed after Nico Rosberg accused Lewis Hamilton of pulling off such a manoeuvre in China.

Figure 5.11: Example from CNN/Daily Mail test dataset showing outputs for the models using reinforcement-based learning (e.g. RL), maximum likelihood training objective (e.g. ML), and mixed objective function (e.g. ML+RL). The ROUGE scores correspond to a specific example. Source: [63]

Input Document (truncated):
beijing cnn chinese police on monday released five female activists who were detained last month , family and friends of the women tell cnn . wei tingting , wang man , zheng churan , li tingting and wu rongrong were freed . the women will be under police surveillance for a year and have their movements and activities restricted , attorney liang xiaojun said . police can summon the women for questioning at any time , he added . the five members of china 's women 's rights action group were detained in beijing , guangzhou and hangzhou a few days before events planned for international women 's day on march 8 . the united states had urged china to free them , and the international community harshly criticized keeping the women in custody . (....)

Reference Summary:
wei tingting , wang man , zheng churan , li tingting and wu rongrong are free .
they will be under police surveillance for a year and have their activities restricted , attorney says .
the international community has harshly criticized keeping the women in custody .

Pointer Generator Model:
five female activists detained in beijing , guangzhou and hangzhou a few days before events planned for international women 's day on march 8 .
the united states had urged china to free them , and the international community harshly criticized keeping the women in custody .
`` free the five '' became a twitter hashtag .

Unified Model :
chinese police on monday released five female activists who were detained last month .
the women will be under police surveillance for a year and have their movements and activities restricted .
the five members of china 's women 's rights action group were detained in beijing , guangzhou and hangzhou a few days before events planned for international women 's day on march 8 .

Our End-to-End Model:
wei tingting , wang man , zheng churan , li tingting and wu rongrong were freed .
the women will be under police surveillance for a year and have their movements and activities restricted .
the united states had urged china to free them , and the international community harshly criticized keeping the women in custody .

Figure 5.12: An example from the test set comparing generated summaries from pointer-generator networks, unified model, and our end-to-end model.

Input Document (truncated):
cnn mercedes driver and f1 championship leader lewis hamilton stole pole position for sunday 's chinese grand prix from teammate and fierce rival nico rosberg in dramatic fashion . hamilton took first place on the front row on the last lap , beating rosberg by a slim four hundredths of a second margin . frenemies the two former friends have enjoyed , or rather endured , a heated rivalry since falling out last season and rosberg 's annoyance at hamilton 's last ditch success was obvious . the german appeared upset as he left his car and refused to shake hamilton 's hand . he did , however , find time to congratulate fellow german driver sebastian vettel , who will start in third after the ferrari driver surprisingly won the malaysian gp two weeks ago . `` i do n't actually mind him having more poles , '' a nonplussed hamilton told afp after the session . `` he can have the trophy for most poles . as long as i have the trophy for winning that 's all that matters . ultimately qualifying is not the end of the world . '' `` i am frustrated '' in the post qualifying press conference rosberg appeared more magnanimous , but still shocked by hamilton 's last minute burst of speed . (...)

Reference Summary:
lewis hamilton will start on pole in china .
pushed teammate rosberg into second on the last lap .
rosberg refused to shake hamilton 's hand afterward .

Pointer Generator Model:
lewis hamilton takes first place on the front row on the last lap .
the german appeared upset as he left his car and refused to shake hamilton 's hand .
the two former friends have enjoyed , or rather endured , a heated rivalry .

Unified Model:
mercedes driver and f1 championship leader lewis hamilton stole pole position for sunday 's chinese grand prix from teammate nico rosberg .
hamilton took first place on the front row on the last lap , beating rosberg by a slim four hundredths of a second margin .
the two former friends have enjoyed , or rather endured , a heated rivalry since falling out last season and rosberg 's annoyance at hamilton 's last ditch success was obvious .

Our End-to-End Model:
mercedes driver lewis hamilton stole pole position for sunday 's chinese grand prix .
hamilton took first place on the last lap , beating rosberg by a slim four hundredths of a second margin .
german appeared upset as he left his car and refused to shake hamilton 's hand .

Figure 5.13: The second example from the test set comparing generated summaries from pointer-generator networks, unified model, and our end-to-end model.

Input Document (truncated):
tokyo cnn a second robotic probe sent into the crippled fukushima nuclear plant has captured images of a strange green glow . tokyo electric power company tepco deployed the second remote-controlled robot last week after the first one broke down . the robot detected lower radiation levels and temperature than expected , an indicator that cooling systems were working effectively , according to a statement released by tepco . `` it is a great step forward towards the decommissioning work as we can earn necessary data for the next investigation , '' said akira ono , the chief of fukushima daiichi plant . tepco said the yellow seen on the images seemed to suggest a discoloration of the grating , though the cause was unknown . it said the green glow could not be seen when filmed from other angles . the shape-shifting robots were sent in to assess the damage in one of the reactors that suffered a meltdown after a devastating earthquake and tsunami in march 2011 . (...)

Reference Summary:
a robotic probe into the fukushima nuclear plant released crucial information on conditions inside the reactor .
tepco : recorded radiation levels and temperatures are lower than expected .
the robot was sent into the plant after the first one broke down .

Pointer Generator Model:
tokyo electric power company deployed the second remote-controlled robot last week .
the robot detected lower radiation levels and temperature than expected .
an indicator that cooling systems were working effectively , according to a statement .

Unified Model:
tokyo electric power company deployed the second remote-controlled robot last week .
the robot detected lower radiation levels and temperature than expected , according to a statement .
tepco said the yellow seen on the images seemed to suggest a discoloration of the grating .
the shape-shifting robots were sent in to assess the damage in one of the reactors that suffered a meltdown after a devastating earthquake and tsunami in march 2011 .

Our End-to-End Model:
tokyo electric power company deployed the second remote-controlled robot last week .
the robot detected lower radiation levels and temperature than expected .
tepco says the site 's still too dangerous for workers to enter .
the first robot , which was sent in on april 10 , stalled after moving about 10 meters .

Figure 5.14: The third example from the test set comparing generated summaries from pointer-generator networks, unified model, and our end-to-end model.

Input document (truncated):
in 140 characters or less . manning , who is serving a 35-year prison sentence for leaking thousands of classified documents , appears to have joined twitter this week . in a series of tweets , the prisoner formerly known as bradley manning said she will be using a voice phone to dictate her tweets to communications firm fitzgibbon media , which will post them on her behalf . she is not allowed internet access in prison , according to the guardian . `` it will be hard , but i do n't want this twitter feed to be a one-way street/conversation , '' manning posted to her nearly 26,000-plus followers . manning was sentenced in 2013 , and in august of that year , she said she wanted to transition to a female . (...)

Reference summary:
manning is serving a 35-year sentence for leaking thousands of classified documents .
she says she will be using a voice phone to dictate her tweets .

Our end-to-end model:
manning , who is serving a 35-year prison sentence for leaking thousands of classified documents , appears to have joined twitter this week .
in a series of tweets , the prisoner formerly known as bradley manning said she will be using a voice phone to dictate her tweets to communications firm fitzgibbon media .
manning was sentenced in 2013 , and in august of that year , she said she wanted to transition to a female .

Input document (truncated):
cnn seventy years ago , anne frank died of typhus in a nazi concentration camp at the age of 15 . just two weeks after her supposed death on march 31 , 1945 , the bergen-belsen concentration camp where she had been imprisoned was liberated timing that showed how close the jewish diarist had been to surviving the holocaust . but new research released by the anne frank house shows that anne and her older sister , margot frank , died at least a month earlier than previously thought . researchers re-examined archives of the red cross , the international training service and the bergen-belsen memorial , along with testimonies of survivors . (...)

Reference summary:
museum : anne frank died earlier than previously believed .
researchers re-examined archives and testimonies of survivors .
anne and older sister margot frank are believed to have died in february 1945 .

Our end-to-end model:
anne frank died of typhus in a nazi concentration camp at the age of 15 .
just two weeks after her supposed death on march 31 , 1945 , the bergen-belsen concentration camp .
but new research released by the anne frank house shows anne and her older sister , margot frank , died at least a month earlier than previously thought .
researchers re-examined archives of the red cross , the international training service and the bergen-belsen memorial , along with testimonies of survivors .

Figure 5.15: Test-set examples where summaries from our model are much less abstractive than the reference summaries. The input parts that are used to generate the final system summary are highlighted in red

Chapter 6.

Conclusion

We introduced an end-to-end neural network framework to generate multiple-sentence summaries from long newswire articles covering various topics using the challenging dataset of CNN/Daily Mail. Our experimental results and analysis demonstrate the effectiveness of combining selector and abstractor models to take advantage of the strengths of both the approaches. While the hierarchical model helps in identifying the important parts in an input document, the pointer generator networks allow the final summaries to be abstractive, consisting of novel words, similar to human-written summaries. Our findings also suggest that training the models jointly in an end-to-end way by minimizing the overall loss is more effective than training the models separately. Qualitative analysis shows that the generated summaries are informative as well as fluent. On this basis, we can say that the selector-abstractor combined architecture is a promising direction for the automatic abstractive summarization task.

For evaluation, we used intrinsic automatic evaluation metrics like ROUGE and METEOR which rely mostly on the content overlap between system-reference words and do not account for other quality aspects such as grammaticality and coherence. As discussed in Section 5.2.1 of Chapter 5, a generated abstractive summary could express the same content as a reference summary without any lexical overlap but still be scored lower by these metrics, unlike how humans would rate the summaries. Moreover, it is preferred to use multiple reference summaries rather than a single person's judgment to evaluate the generated summaries, but the dataset we used for the experiments is limited to a single gold-standard summary for each input document. Hence, to justify the quality of the generated summaries additional extrinsic human-based evaluation is needed, where one option is to use a set of questions and let several human evaluators score summaries on a pre-set scale (e.g. 1-5) based on different linguistic aspects for the task at hand like:

Novelty: How abstractive is the summary consisting of novel phrases?
Fluency: Is the summary grammatically correct and readable?

Informativeness: Does the summary capture all the important information from the input document?

Conciseness: Does the summary provide a gist of the document in a concise manner?

Redundancy: Is the information in the summary repetitive?

As an extension of this research work, we plan to train the end-to-end model by improving the intermediate modules. From *Figure 5.1*, we can see that including previous summary information to inform the next sentence predictions did not make much improvement over other features. One reason for this might be since predicting the probability of selecting the next sentence uses information from all previous timesteps, the selector model may not be able to remember long-range information for timesteps further away. To handle this shortcoming of the current selector, an option is to predict the selection probabilities using an attentional decoder, which would allow the model to remember previous decisions at all timesteps. This change might improve the overall performance, but it needs to be validated with further experiments.

Additionally, as shown in Figure *5.15*, our current abstractor is limited to producing a smaller number of novel phrases in the summaries compared to the reference summaries. One option is to utilize a pretrained language model, which is trained on a large external news corpus, like NEWSROOM corpus where the articles come from diverse newswire sources and the summaries combine extractive and abstractive styles, to incorporate prior knowledge about language generation in the decoder to help in generating more novel phrases.

Bibliography

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

[2] C.D. Santos and B. Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826, 2014.

[3] M. Sundermeyer, R. Schluter, and H. Ney. LSTM neural networks for language modeling. In Thirteenth Annual Conference of the International Speech Communication Association, pages 194–197, 2012.

[4] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

[5] E. Sandhaus. The New York Times annotated corpus. *Linguistic Data Consortium LDC2008T19*, DVD, Philadelphia, 2008. URL https://catalog.ldc.upenn.edu/LDC2008T19

[6] M. Grusky, M. Naaman, and Y. Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*, 2018.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, volume 9, Issue 8, pages 1735–1780, 1997.

[8] R. Nallapati, F. Zhai, and B. Zhou. SummaRuNNer: A recurrent neural network-based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3075–3081, 2017.

[9] A. Nenkova. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, Pittsburgh, PA., volume 5, pages 1436–1441, 2005.

[10] R. Nallapati, B. Zhou, C.N. dos Santos, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.

[11] S. Chopra, M. Auli, and A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL-HLT16*, pages 93–98, 2016.

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

[13] G.A. Miller, R. Beckwith, C.D. Fellbaum, D. Gross, and K. Miller. WordNet: An online lexical database. *International Journal of Lexicography*, volume 3, pages 235–244, 1990.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[15] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, volume 12, pages 2121–2159, 2011.

[16] 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. *arXiv preprint arXiv:1805.06266*, 2018.

[17] I. Mani. Automatic Summarization. *John Benjamins Publishing Company*, 2001.

[18] J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 484–494, 2016.

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

[20] B. Dorr, D. Zajic, and R. Schwartz. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 2003 Text Summarization Workshop*, Edmonton, Alberta, Canada, pages 1–8, 2003.

[21] 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.

[22] C.Y. Lin and E. Hovy. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 17th Conference on Computational Linguistics*, volume 1, pages 495–501, 2000.

[23] K. Owczarzak and H.T. Dang. TAC 2010 guided summarization task guidelines. In *Proceedings of the Text Analysis Conference*, 2010. URL https://tac.nist.gov//2010/Summarization/index.html

[24] P.F. Brown, S.A.D. Pietra, V.J.D. Pietra, and R.L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, volume 19, pages 263–312, 1993.

[25] P.F. Brown, J. Cocke, S.A.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roosin. A statistical approach to machine translation. *Computational Linguistics*, volume 16, pages 79–85, 1990.

[26] M. Banko, V.O Mittal, and M.J Witbrock. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325, 2000.

[27] G.D. Forney. The Viterbi algorithm. In *Proceedings of the IEEE*, pages 268–278, 1973.

[28] D. Zajic, B. Dorr, and R. Schwartz. Automatic headline generation for newspaper stories. In *Workshop on Automatic Summarization*, Philadelphia, PA, pages 78–85, 2002.

[29] Harman, Donna, and M. Liberman. TIPSTER Complete. Linguistic Data Consortium LDC93T3A, DVD, Philadelphia, 1993. URL https://catalog.ldc.upenn.edu/LDC93T3A

[30] C.Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, Barcelona, Spain, pages 74–81, 2004.

[31] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, pages 311–318, 2002.

[32] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013.

[33] F. Liu, J. Flanigan, S. Thomson, N. Sadeh, and N.A. Smith. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado, pages 1077–1086, 2015.

[34] J. Flanigan, S. Thomson, J. Carbonell, C. Dyer, and N.A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistic (ACL)*, pages 1426–1436, 2014.

[35] S. Dohare and H. Karnick. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*, 2017.

[36] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume 35, pages 400–401, 1987.

[37] D. Jurafsky and J.H. Martin. Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics. 2nd edition. *Prentice Hall*, 2009.

[38] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, volume 5, pages 103–233, 2011.

[39] Wikipedia contributors. Artificial neuron. In *Wikipedia, The free encyclopedia*. URL https://en.wikipedia.org/w/index.php?title=Artificial_neuron&oldid=912587183

[40] M. Nielsen. Neural networks and deep learning. URLhttp://neuralnetworksanddeeplearning.com/chap1.html

[41] D. Britz. Recurrent neural networks tutorial, part 1- Introduction to RNNs. URL http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[42] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by backpropagating errors. *Nature 323*, pages 533–536, 1986.

[43] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[44] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, volume 3, pages 1137–1155, 2003.

[45] D. Graff, J. Kong, K. Chen, and K. Maeda. English Gigaword. *Linguistic Data Consortium LDC2003T05*, Philadelphia, 2003. URL https://catalog.ldc.upenn.edu/LDC2003T05

[46] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, volume 5, pages 157–166, 1994.

[47] F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194, 2000.

[48] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*, 2015.

[49] R. Nallapati, B. Xiang, and B. Zhou. Sequence-to-sequence RNNs for text summarization. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.

[50] 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 (EMNLP)*, pages 1054–1059, 2016.

[51] Y. Miao and P. Blunsom. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 319–328, 2016.

[52] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1–10, 2015.

[53] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer Networks. *arXiv preprint arXiv:1506.03134*, 2015.

[54] D. Matthews, T. Behne, E. Lieven, and M. Tomasello. Origins of the human pointing gesture: A training study. *Developmental Science*, volume 15, pages 817–829, 2012.

[55] M. Tomasello, M. Carpenter, and U. Liszkowski. A new look at infant pointing. *Child development*, volume 78, pages 705–722, 2007.

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

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

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

[59] A. See, P.J. Liu, and C.D. Manning. Get to the point: Summarization with pointer generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1073–1083, 2017.

[60] C. Li, W. Xu, S. Li, and S. Gao. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 55–60, 2018.

[61] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *MIT Press*, USA, 1998.

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

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

[64] W. Kryscinski, R. Paulus, C. Xiong, and R. Socher. Improving abstraction in text summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1808–1817, 2018.

[65] S.J Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*, 2016.

[66] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li. Generative adversarial network for abstractive text summarization. *arXiv preprint arXiv:1711.09357*, 2017.

[67] R. Pasunuru and M. Bansal. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 646–653, 2018.

[68] Y.C. Chen and M. Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*, 2018.

[69] J. Pennington, R. Socher, and C.D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[70] C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[71] H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, volume 2, pages 159–165, 1958.

[72] K.M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 1, pages 1693–1701, 2015.

[73] D. Zajic, B. Dorr, and R. Schwartz. BBN/UMD at DUC-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop*, Boston, pages 112–119, 2004.